z/VM

TCP/IP User's Guide

Version 6 Release 4

SC24-6240-07

Note:
Before you use this information and the product it supports, read the information in "Notices" on page 507.

This edition applies to version 6, release 4, modification 0 of IBM z/VM (product number 5741-A07) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-6240-06.

© Copyright IBM Corporation 1987, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures xi

Tables xiii

About This Document xv
Intended Audience xv
Conventions and Terminology xv
How the Term "internet" Is Used in This Document. xv
How Numbers Are Used in This Document xv
Syntax, Message, and Response Conventions xvi
Where to Find More Information xviii
Links to Other Documents and Websites xviii

How to Send Your Comments to IBM xix

Summary of Changes xxi
SC24-6240-07, z/VM Version 6 Release 4 xxi
z/VM LDAP Server and Client Utilities z/OS V2.2 Equivalency xxi
zManager Support Removed xxi
z/VM System SSL and Utilities z/OS V2.2 Equivalency xxi
Domain Name System (DNS) IPv6 Support xxi
SC24-6240-06, z/VM Version 6 Release 3 (September 2015) xxi
System SSL z/OS 2.1 Equivalency xxii
SC24-6240-05, z/VM Version 6 Release 3 (June 2014) xxii
Certificate (Key) Database Administration Information xxii
SC24-6240-04, z/VM Version 6 Release 3 xxii
IPv6 Support for SSL. xxii
Removal of A220, CLAW, DHCP and LPD Support xxii
TLS 1.2 Support xxii
SC24-6240-03, z/VM Version 6 Release 2 (Updated Edition) xxii
Virtual Switch HiperSockets Bridge xxii
Client Certificate Validation for z/VM SSL Server xxiii
SC24-6240-02, z/VM Version 6 Release 2 xxiii
IPv6 Support for FTP Server and Client xxiii
IPv6 Support for SMTP Server and Client xxiii
GSKKYMAN xxiii
MPROUTE Enhancements and Upgrade to z/OS 1.12 Equivalency xxiii
LDAP Enhancements xxiii
Query OSA Address Table. xxiv
Support Withdrawal. xxiv
SC24-6240-01, z/VM Version 6 Release 1 (Updated Edition) xxiv
z/VM SSL Server Enhancements xxiv
Support for IBM zEnterprise System Networks xxiv
SC24-6240-00, z/VM Version 6 Release 1 xxiv
GSKKYMAN and GSKTRACE xxv
NSLOOKUP. xxv

Chapter 1. Introducing Computer Networks and Protocols 1
Computer Networks. 1
Internet Environment 1
Internet Protocol Version 4 and Internet Protocol Version 6 3
TCP/IP Protocols and Functions. 5
Link Protocols 6
Network Protocols 7

© Copyright IBM Corp. 1987, 2016

Transport Protocols . . . 9
Applications and Protocols . . . 9
Routing . . . 13
Internet Addressing . . . 13
IPv4 Addressing . . . 13
IPv6 Addressing . . . 16

Chapter 2. Transferring Files Using FTP . . . 27
FTP Command . . . 28
The NETRC DATA File . . . 31
The FTP DATA File . . . 31
Statement Syntax . . . 32
FTP DATA File Statements . . . 32
CCONNTIME Statement . . . 33
DATACTTIME Statement . . . 34
DCONNTIME Statement . . . 35
EPSV4 Statement . . . 36
FTPKEEPALIVE Statement . . . 37
INACTTIME Statement . . . 38
LOADDBCSTABLE Statement . . . 39
MYOPENTIME Statement . . . 40
SECURECONTROL Statement . . . 41
SECUREDATA Statement . . . 42
FTP Subcommands . . . 43
Continuing Subcommand Input Strings . . . 45
File Name Formats . . . 46
Fully-Qualified Pathnames . . . 49
Interpretation of Path Information . . . 50
Storage Space Requirements for Transferred Files . . . 51
File Transfer Methods . . . 51
File List Formats . . . 53
Transferring Files Using a Web Browser . . . 56
Transferring Files Using Secure FTP . . . 57
ACCT . . . 58
APPEND . . . 59
ASCII . . . 59
BINARY . . . 60
CCC . . . 61
CD or CWD . . . 62
CDUP . . . 65
CLEAR . . . 65
CLOSE . . . 66
CMS . . . 66
CPROTECT . . . 67
DEBUG . . . 67
DELETE . . . 68
DELIMIT . . . 68
DIR . . . 69
EBCDIC . . . 70
EUCKANJI . . . 71
GET . . . 71
HANGEUL . . . 72
HELP . . . 73
IBMKANJI . . . 73
JIS78KJ . . . 74
JIS83KJ . . . 74
KSC5601 . . . 75
LCD . . . 75
LOCSITE . . . 76
LOCSTAT . . . 77
LPWD . . . 78

LS . . . 78
MDELETE . . . 80
MGET . . . 81
MKDIR . . . 81
MODE . . . 82
MPUT . . . 83
NETRC . . . 84
NOOP . . . 84
OPEN . . . 85
PASS . . . 86
PASSIVE . . . 86
PRIVATE . . . 87
PUT . . . 88
PWD . . . 89
QUIT . . . 90
QUOTE . . . 90
RENAME . . . 91
RMDIR . . . 91
SENDPORT . . . 92
SENDSITE . . . 93
SITE . . . 93
SIZE . . . 96
SJISKANJI . . . 96
STATUS . . . 97
STRUCT . . . 97
SUNIQUE . . . 98
SYSTEM . . . 99
TCHINESE . . . 100
TYPE . . . 100
USER . . . 101
Using FTP Within an EXEC . . . 102
Providing FTP Subcommand Input . . . 103
Managing FTP Command Output . . . 103
Examples . . . 103
FTP Return Codes . . . 105
Examples . . . 107
FTP Reply Codes . . . 107
Internal Error Codes . . . 109
Using FTP with RACF . . . 110

Chapter 3. Transferring Files Using TFTP . . . 111
TFTP Command . . . 111
Creating Translation Tables . . . 112
TFTP Subcommands . . . 112
GET . . . 112
HELP . . . 113
LOCSTAT . . . 113
MAXPKT . . . 114
MODE . . . 114
OPEN . . . 115
PUT . . . 115
QUIT . . . 116
REXMIT . . . 116
TRACE . . . 116

Chapter 4. Sending and Receiving Electronic Mail . . . 119
NOTE and SENDFILE Commands . . . 119
Electronic Mail Gateway . . . 119
Note and File Delivery . . . 120
SMTPQUEU Command . . . 120

Undelivered Notes . . . 121
Nondelivery Note . . . 121
Unknown Recipient Note . . . 122
Using OfficeVision Without OfficeVision Extended Mail Installed . . . 122
Specifying TCP/IP Recipients . . . 122
Example of Sending a Note Copy to SMTP . . . 123
Note Delivery . . . 124
Using IMAP to Access Electronic Mail . . . 124
Using Secure SMTP . . . 124
Using the SMSG Interface to SMTP . . . 125
General User SMSG Commands . . . 125
Receiving Electronic Mail on VM . . . 127

Chapter 5. Logging On to a Foreign Host . . . 129
TELNET Command . . . 129
TELNET Function Keys . . . 131
In Transparent (TN3270) Mode . . . 131
In Line Mode . . . 131
TELNET Subcommands . . . 132
AO . . . 132
AYT . . . 132
BRK . . . 133
HELP . . . 133
IP . . . 133
PA1 . . . 134
QUIT . . . 134
SYNCH . . . 135
Sending ASCII Control Characters to a Host in Line Mode . . . 135

Chapter 6. Using the LDAP Client Utilities . . . 137
Running the LDAP Client Utilities in CMS . . . 137
Using the Command Line Utilities . . . 138
Use of BFS Files . . . 138
Using Special Characters with the LDAP Utilities . . . 139
Specifying a value for a filename . . . 139
SSL/TLS information for LDAP utilities . . . 139
SSL initialization failure . . . 140
Using Environment Variables to Control SSL/TLS Settings . . . 142
Enabling Tracing . . . 143
LDAP URLs and LDAP Filters . . . 144
LDAPCHPW (ldapchangepwd Utility) . . . 146
LDAPCMPR (ldapcompare Utility) . . . 149
LDAPDLET (ldapdelete utility) . . . 153
LDAPMDFY and LDAPADD (ldapmodify and ldapadd Utilities) . . . 156
LDAPMRDN (ldapmodrdn Utility) . . . 169
LDAPSRCH (ldapsearch Utility) . . . 174

Chapter 7. Monitoring the TCP/IP Network . . . 183
NETSTAT Displaying Local Host Information . . . 183
NETSTAT Command Address Interpretation . . . 183
NETSTAT Command . . . 184
Examples . . . 202
RPCINFO Command . . . 224
TRACERTE Command . . . 225
PING Command . . . 228

Chapter 8. SSL Certificate/Key Management and SSL Tracing Information . . . 233
SSL Certificate Management . . . 233
gskkyman Overview . . . 234
Key Database Files . . . 235

PKCS #12 Files . . . 235
Elliptic Curve Cryptography Support . . . 235
gskkyman Interactive Mode Descriptions . . . 236
GSKKYMAN Interactive Mode Examples . . . 247
GSKKYMAN Command Line Mode Syntax . . . 285
Certificate (Key) Database Administration . . . 291
Creating and Using a Key Database . . . 292
Obtaining a Certificate . . . 292
Using Self-Signed Certificates . . . 293
Certificate Management . . . 293
SSL Tracing Information . . . 293
GSKTRACE (gsktrace utility) command . . . 294

Chapter 9. Using the Network File System Commands. . . . 297
VM's NFS Server Support . . . 297
NFS Client Commands . . . 298
MOUNT Command . . . 298
MOUNTPW Command . . . 308
UMOUNT Command . . . 309
Diagnosing Mount Problems . . . 310
Errors Using Mounted Systems . . . 311
Notes for BFS Files and Directories . . . 313
Notes for SFS Files and Directories . . . 314
Notes for Minidisk Files . . . 316
SMSG Interface to VMNFS . . . 318
SMSG DETACH Command. . . . 318
SMSG ERROR Command . . . 319
SMSG QUERY Command . . . 320
SMSG REFRESH user_id.vaddr Command . . . 323
Name Translation File . . . 324
Special File Names for VM NFS . . . 325
Special File Name Considerations . . . 325
NFS Client Problems . . . 325
Deleting CMS Record-Length Fields . . . 326
Using NFS with RACF . . . 326
VM NFS Server Link Support . . . 326
SFS and Minidisk Links . . . 327
BFS Links . . . 327

Chapter 10. Using the Remote Execution Protocol . . . 329
REXEC Command. . . . 329
The NETRC DATA File . . . 331
Anonymous Remote Command Execution. . . . 331
Command Execution Using Your Own Virtual Machine . . . 332
Notes and Restrictions . . . 332
Using RSH commands with REXECD . . . 333

Chapter 11. Using Remote Printing . . . 335
LPRSET Command . . . 335
LPR Command. . . . 339
Controlling LPSERVE from other Systems . . . 352
LPQ Command. . . . 353
LPRM Command . . . 355

Chapter 12. Using GDDMXD/VM with the X Window System . . . 357
Overview of GDDMXD/VM . . . 357
GDDM Application Limitations . . . 357
GDDM Display Limitations . . . 357
z/VM Considerations . . . 358
GDDMXD Command. . . . 358

Configuring the GDDMXD/VM Environment . . . 359
Identifying the Target Display . . . 359
GLOBALV Command . . . 359
User-Specified Commands . . . 360
Resizing the GDDMXD Graphics Window . . . 360
gddmx*Geometry: Command . . . 361
gddmx*GColornn: Command . . . 362
gddmx*GMCPnn: Command . . . 363
gddmx*ZWL: Command . . . 363
gddmx*XSync: Command . . . 364
gddmx*CMap: Command . . . 364
gddmx*HostRast: Command . . . 365
gddmx*XClConn: Command . . . 366
gddmx*Compr: Command . . . 366
gddmx*PDTrace: Command . . . 367
gddmx*ANFontn Command . . . 367
Remapping the Enter and Newline Keys . . . 368
gddmx*Enter: Command . . . 368
gddmx*NewLine: Command . . . 369
GDDMXD/VM Keyboard Functions . . . 369
GDDMXD/VM to X Window System Keyboard Functions . . . 370
APL2 Character Set Keyboard . . . 370
Setting Up GDXAPLCS MAP . . . 371
Collecting Problem Determination Information . . . 371
GDDMXD/VM—X Window System Server Environment . . . 371
Obtaining the GDDM Trace . . . 371
Obtaining the GDDMXD Problem Determination Trace . . . 372
GDDMXD/VM Problem Determination Examples . . . 372
Problem: Messages from the X Window System or from GDDMXD/VM indicating the connection could not be established. . . . 372
Problem: Any other problem such as program check or ABEND. . . . 372
Problem: An X Window System message indicating an error or any unexpected ABEND. . . . 373
Problem: GDDMXD window is created, but picture drawn is incorrect or user interaction with application produces unexpected results. . . . 373
Demonstration Programs . . . 373
GXDEMO1 . . . 373
GXDEMO2 . . . 373
GXDEMO3 . . . 374
GXDEMO4 . . . 374
GXDEMO4A . . . 374
GXDEMO5 . . . 374
GXDEMO6 . . . 374

Chapter 13. Managing TCP/IP Network Resources with SNMP . . . 375

Sample Command Lists . . . 375
SNMP/NetView Overview . . . 376
SNMP Commands . . . 377
SNMP Commands Overview . . . 377
Return Codes . . . 377
SNMP GET Command . . . 378
SNMP GETNEXT Command . . . 380
SNMP SET Command . . . 382
SNMP TRAPSON Command . . . 384
SNMP TRAPSOFF Command . . . 386
SNMP MIBVNAME Command . . . 387
SNMP PING Command . . . 388
SNMP Native Commands . . . 389
SNMPTRAP Command . . . 390
Major and Minor Error Codes and SNMP Value Types . . . 393
gethostbyname() . . . 394

Chapter 14. Using the Domain Name System . . . 395
Overview of the Domain Name System . . . 395
Domain Names . . . 395
Domain Name Servers . . . 396
Resolvers . . . 397
Resource Records . . . 398
NSLOOKUP—Querying Name Servers . . . 401
NSLOOKUP Internal State Information . . . 401
NSLOOKUP Command . . . 401
NSLOOKUP Interactive Session Subcommands . . . 404
NSLOOKUP set Subcommands . . . 409
NSLOOKUP Examples . . . 419
DIG—Querying Name Servers . . . 421
DIG Internal State Information . . . 422
DIG Command . . . 422
DIG Examples . . . 428
CMSRESOL—Resolver and Name Server . . . 433
RESOLV Command—Interface to the CMSRESOL MODULE . . . 433
CMSRESOL Command—Interface to the Resolver . . . 434
Types of Queries . . . 435
The Standard Query . . . 435
The Inverse Query . . . 435
Querying the in-addr.arpa Domain . . . 435
Querying Outside Your Domain . . . 436

Chapter 15. Using Translation Tables . . . 437
Character Sets and Code Pages . . . 437
TCP/IP Translation Table Files . . . 438
Translation Table Search Order . . . 438
Special Telnet Requirements . . . 439
IBM-Supplied Translation Tables . . . 440
Customizing SBCS Translation Tables . . . 442
Syntax Rules for SBCS Translation Tables . . . 443
Customizing DBCS Translation Tables . . . 443
DBCS Translation Table . . . 444
Syntax Rules for DBCS Translation Tables . . . 444
Sample DBCS Translation Tables . . . 444
Converting Translation Tables to Binary . . . 446
CONVXLAT Command . . . 447

Appendix A. Specifying Files and Data Sets . . . 449
AIX Files . . . 449
OS/390 Data Sets . . . 450
Sequential Data Sets . . . 450
Partitioned Data Sets . . . 450
DOS, OS/2, and Windows Files . . . 451
AS/400 Operating System . . . 452

Appendix B. Using the NETRC DATA File . . . 453
Using The NETRC DATA File with FTP . . . 453
Using The NETRC DATA File with REXEC . . . 454
Using the NETRC DATA File with the OPEN MOUNT CMS Command . . . 454

Appendix C. Mapping Values for the APL2 Character Set . . . 455

Appendix D. Using DBCS with FTP and Mail . . . 461
Using DBCS with FTP . . . 461
DBCS Translation Tables . . . 461
DBCS Subcommands . . . 461

Defining FTP with Kanji Support . . . 464
Using FTP with Kanji Support . . . 464
Using DBCS with Mail . . . 465
SMTP Server DBCS Support . . . 465
SMTP Protocol for 8-bit characters . . . 465
Conversion of DBCS Mail . . . 466
Conversion of DBCS Files . . . 466

Appendix E. Management Information Base Objects . . . 467
Structure and Identification of Management Information (SMI) . . . 467
Names . . . 468
The Management Subtree . . . 468
MIB/Network Elements . . . 470
System Group . . . 471
Interfaces Group . . . 473
Address Translation Group . . . 478
IP Group . . . 478
IP Address Translation Table . . . 483
ICMP Group . . . 484
TCP Group . . . 486
UDP Group . . . 489
Bridge Group . . . 489

Appendix F. SNMP Generic TRAP Types . . . 495

Appendix G. Related Protocol Specifications . . . 497

Appendix H. Abbreviations and Acronyms . . . 503

Notices . . . 507
Programming Interface Information . . . 509
Third Party Copyright Information . . . 509
Trademarks . . . 512
Terms and Conditions for Product Documentation . . . 513
IBM Online Privacy Statement . . . 514

Glossary . . . 515

Bibliography . . . 517
Where to Get z/VM Information . . . 517
z/VM Base Library . . . 517
z/VM Facilities and Features . . . 518
Prerequisite Products . . . 519
Other TCP/IP Related Publications . . . 519

Index . . . 521

Figures

1. The TCP/IP Layered Architecture . . . 6
2. Class A Address . . . 14
3. Class B Address . . . 14
4. Class C Address . . . 14
5. Class D Address . . . 14
6. Class B Address with Subnet . . . 16
7. Link-local scope zones . . . 20
8. Site-local scope zones . . . 21
9. Flags in multicast address . . . 23
10. Example SMTP Mail Gateway Environment . . . 119
11. Example of a Nondelivery Note . . . 121
12. Example of an Unknown Recipient Note . . . 122
13. Database Menu . . . 237
14. Key Management Menu . . . 239
15. Key and Certificate Menu . . . 240
16. Certificate Menu . . . 242
17. Request Menu . . . 243
18. Starting Menu for GSKKYMAN . . . 248
19. Creating a New Key Database . . . 248
20. Key Management Menu for GSKKYMAN . . . 249
21. Opening an Existing Key Database File . . . 250
22. Key Management Menu . . . 250
23. Deleting an Existing Key Database . . . 251
24. Changing a Key Database Password . . . 252
25. Storing a Database Password in a Stash File . . . 253
26. Select 6 to Create a Self-Signed Certificate . . . 254
27. Creating a Self-Signed Certificate . . . 255
28. Select 4 to Create a New Certificate Request . . . 257
29. Creating a Certificate Request . . . 258
30. Subject Alternate Name Type . . . 259
31. Contents of certreq.arm after Certificate Request Generation . . . 260
32. Receiving a Certificate Issued for your Request . . . 261
33. Key and Certificate List . . . 262
34. Key and Certificate Menu . . . 262
35. Certificate Information . . . 263
36. Certificate Extensions List . . . 263
37. Key Usage Information . . . 264
38. Key Information menu . . . 264
39. Marking a Certificate (and Private Key) as the Default Certificate . . . 265
40. Copying a Certificate Without its Private Key . . . 266
41. Copying a Certificate and Private key to a Different Key Database . . . 267
42. Copying a Certificate with its Private Key to a Key Database on the Same System . . . 268
43. Delete Certificate and Key . . . 269
44. Changing a Certificate Label . . . 270
45. Creating a Signed Certificate and Key . . . 270
46. Certificate Type menu . . . 272
47. Subject Alternate Name menu . . . 273
48. Selecting the ECC Key Type . . . 274
49. Selecting the ECC Curve Type . . . 275
50. Creating a key parameter file to be used with Diffie-Hellman . . . 276
51. Creating a certificate to be used with Diffie_Hellman . . . 277
52. Select 11 to Create a Certificate Renewal Request . . . 278
53. Certificate List (part 1) . . . 279
54. Certificate List (part 2) . . . 280
55. Certificate List (part 3) . . . 280

© Copyright IBM Corp. 1987, 2016

56. Importing a Certificate from a File 281
57. Importing a Certificate and Private Key from a File 282
58. NFS MOUNT Command 305
59. NFS UMOUNT Command 310
60. Overview of NetView SNMP Support 376
61. Hierarchical Tree 396
62. A TCP/IP Network 419
63. The SMI Hierarchical Tree 469

Tables

1. Examples of Syntax Diagram Conventions . . . xvi
2. Functional Groups . . . 5
3. IPv6 Address Format . . . 17
4. Types of IPv6 Addresses . . . 18
5. Unicast Address Format . . . 19
6. Global Unicast Address Format . . . 19
7. Link-local address format . . . 19
8. Site-local address format . . . 20
9. IPv4-mapped IPv6 address . . . 22
10. IPv4-compatible IPv6 address . . . 22
11. Multicast address format . . . 23
12. Multicast scope field values . . . 23
13. FTP and Subcommand Functions . . . 27
14. FTP Subcommands . . . 43
15. Recommended Methods of File Transfer . . . 52
16. FTP Command Codes . . . 106
17. FTP Reply Codes . . . 108
18. Internal Error Codes . . . 109
19. TELNET PF Key Functions . . . 131
20. Control Characters . . . 135
21. SSL failure reason codes . . . 140
22. LDAP trace debug levels . . . 143
23. LDAPCHPW options . . . 147
24. LDAPCMPR options . . . 150
25. LDAPDLET options . . . 153
26. LDAPMDFY and LDAPADD options . . . 156
27. LDAPMRDN options . . . 170
28. LDAPSRCH options . . . 174
29. Recommended digest sizes for ECDSA signature key sizes . . . 236
30. Default EC Named Curves for Specified Key Sizes . . . 236
31. File Extension Translation Table . . . 305
32. Executables . . . 309
33. NFS System Return Codes . . . 311
34. Resource Information . . . 323
35. Window Width and Window Size . . . 361
36. GColors . . . 362
37. X Window System Fonts . . . 368
38. Translation Table Files . . . 438
39. Preferred Translation Tables . . . 439
40. IBM Translation Tables . . . 440
41. Mapping Values for the APL2 Character Set . . . 455
42. FTP TYPE commands . . . 463
43. Implementation of the System Group . . . 471
44. Implementation of the Interfaces Group . . . 473
45. Implementation of the Address Translation Group . . . 478
46. Implementation of the IP Group . . . 478
47. IP Address Translation Table . . . 483
48. Implementation of the ICMP Group . . . 484
49. Implementation of the TCP Group . . . 486
50. Implementation of the UDP Group . . . 489
51. Implementation of the Bridge Group - dot1dBase . . . 489

© Copyright IBM Corp. 1987, 2016

About This Document

This document describes the functions of and explains how to use the applications available in TCP/IP running on the IBM® z/VM® V6.4 operating system (TCP/IP level 640). These applications include:

- Transferring files
- Sending electronic mail
- Logging on to a foreign host
- Monitoring the network
- Authenticating network users
- Remote printing
- Managing network resources
- Using the Domain Name System

This document also describes how to use the Network File System (NFS), REXEC command, and GDDMXD with the X Window System.

For information about how to set up, initialize, and customize your TCP/IP, see *z/VM: TCP/IP Planning and Customization* and *z/VM: TCP/IP Programmer's Reference*. For information about error messages that may appear while using TCP/IP, see *z/VM: TCP/IP Messages and Codes*.

Intended Audience

This document is intended for an end-user and describes how to use level 640 after it has been installed and customized on your network. You should read this document when you want to use the applications that are available in level 640.

Before using this document, you should be familiar with the CP and CMS components of z/VM.

In addition, TCP/IP should already be installed and customized for your network.

Conventions and Terminology

This topic describes important style conventions and terminology used in this document.

How the Term "internet" Is Used in This Document

In this document, an internet is a logical collection of networks supported by routers, gateways, bridges, hosts, and various layers of protocols, which permit the network to function as a large, virtual network.

Note: The term "internet" is used as a generic term for a TCP/IP network, and should not be confused with the Internet, which consists of large national backbone networks (such as MILNET, NSFNet, and CREN) and a myriad of regional and local campus networks worldwide.

How Numbers Are Used in This Document

In this document, numbers over four digits are represented in metric style. A space is used rather than a comma to separate groups of three digits. For example, the number sixteen thousand, one hundred forty-seven is written 16 147.

© Copyright IBM Corp. 1987, 2016

Syntax, Message, and Response Conventions

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

How to Read Syntax Diagrams

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

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

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

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in Table 1.

Table 1. Examples of Syntax Diagram Conventions

Syntax Diagram Convention	Example
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►►—KEYWORD————————————►◄
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	►►—KEYWOrd————————————►◄
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	* Asterisk : Colon , Comma = Equal Sign - Hyphen () Parentheses . Period

Table 1. Examples of Syntax Diagram Conventions (continued)

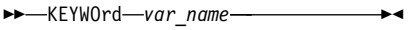
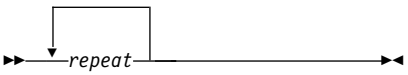
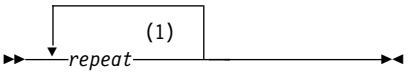
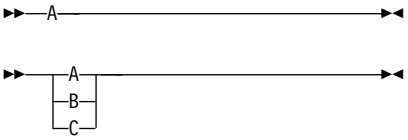
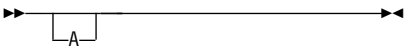
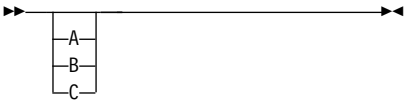
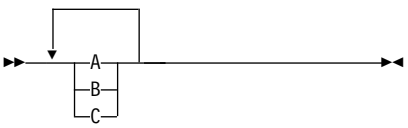
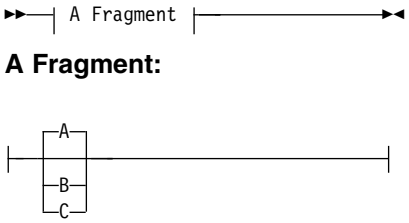
Syntax Diagram Convention	Example
Variables A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	  **Notes:** 1 Specify *repeat* up to 5 times.
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	

Table 1. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	

Examples of Messages and Responses

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

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

[ ] Brackets enclose optional text that might be displayed.

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

| The vertical bar separates items within brackets or braces.

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

Where to Find More Information

Appendix H, "Abbreviations and Acronyms," on page 503, lists the abbreviations and acronyms that are used throughout this document.

The "Glossary" on page 515, defines terms that are used throughout this document associated with TCP/IP communication in an internet environment.

For more information about related publications, see "Bibliography" on page 517.

Links to Other Documents and Websites

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

How to 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. Go to IBM z/VM Reader's Comments (www.ibm.com/systems/z/os/zvm/zvmforms/webqs.html).

Include the following information:

- Your name
- Your email address
- The publication title and order number:
  z/VM V6.4 TCP/IP User's Guide
  SC24-6240-07
- The topic name or 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 use the personal information that you supply only to contact you about the issues that you submit to IBM.

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.
- Contact IBM technical support.
- See IBM: z/VM Service Resources (www.ibm.com/vm/service/).
- Go to IBM Support Portal (www.ibm.com/support/entry/portal/Overview/).

© Copyright IBM Corp. 1987, 2016

Summary of Changes

This document contains terminology, maintenance, and editorial changes. Technical changes are indicated by a vertical line to the left of the change. Some product changes might be provided through service and might be available for some prior releases.

SC24-6240-07, z/VM Version 6 Release 4

This edition includes changes to support the general availability of z/VM V6.4.

z/VM LDAP Server and Client Utilities z/OS V2.2 Equivalency

The LDAP server has been upgraded to the current level of the z/OS® product.

The following information has been changed as a result of this support:

- Chapter 6, "Using the LDAP Client Utilities," on page 137
  - "LDAPMDFY and LDAPADD (ldapmodify and ldapadd Utilities)" on page 156
  - "LDAPSRCH (ldapsearch Utility)" on page 174
- Chapter 8, "SSL Certificate/Key Management and SSL Tracing Information," on page 233

zManager Support Removed

The IBM z™ Unified Resource Manager (zManager) is no longer supported in z/VM V6.3. The vswitch types of IEDN and INMN have been removed from documentation.

z/VM System SSL and Utilities z/OS V2.2 Equivalency

GSKKYMAN can now support RFC 5280 certificates. See "gskkyman Overview" on page 234 for details.

Domain Name System (DNS) IPv6 Support

The TCP/IP support for IPv6 Domain Name Servers updates will support IPv6 domain name system (DNS) servers that have an IPv6 address.

The following information has been changed as a result of this support in Chapter 14, "Using the Domain Name System," on page 395:

- "Resource Records" on page 398
- "NSLOOKUP—Querying Name Servers" on page 401
- "NSLOOKUP Command" on page 401

SC24-6240-06, z/VM Version 6 Release 3 (September 2015)

This edition includes changes to support product changes provided or announced after the general availability of z/VM V6.3.

© Copyright IBM Corp. 1987, 2016

System SSL z/OS 2.1 Equivalency

GSKKYMAN has been enhanced to add support for Elliptic Curve Cryptography (ECC) certificates and keys. See Chapter 8, "SSL Certificate/Key Management and SSL Tracing Information," on page 233.

SC24-6240-05, z/VM Version 6 Release 3 (June 2014)

This edition includes changes to support product changes provided or announced after the general availability of z/VM V6.3.

Certificate (Key) Database Administration Information

Chapter 8, "SSL Certificate/Key Management and SSL Tracing Information," on page 233 has been expanded with information formerly provided in *z/VM: TCP/IP Planning and Customization*.

SC24-6240-04, z/VM Version 6 Release 3

This edition includes changes to support the general availability of z/VM V6.3.

IPv6 Support for SSL

The SSL server has been enhanced to support secure IPv6 SSL connections. The following servers and clients have added support for secure IPv6 connections:

- TELNET server and client
- FTP server and client
- SMTP server

Furthermore, NETSTAT IDENT SSL and SSLADMIN have been enhanced to display secure IPv6 connections.

Removal of A220, CLAW, DHCP and LPD Support

The removal of unsupported TCP/IP device drivers and servers is part of an ongoing effort to provide only those TCP/IP services that are required by customers to support their enterprise.

TLS 1.2 Support

The z/VM SSL Server has been upgraded to exploit V1.13 System SSL, which features Transport Layer Security version 1.2. This expands the cipher suites which can be used by the SSL server and provides increased security. Additionally, support for an SSL configuration PROTOCOL operand is added, which allows an administrator to enable or disable versions of the SSL and TLS protocols.

SC24-6240-03, z/VM Version 6 Release 2 (Updated Edition)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 6.2.

Virtual Switch HiperSockets Bridge

The z/VM virtual switch has been enhanced to transparently bridge a guest virtual machine network connection on a HiperSockets™ LAN segment. This bridge allows a single HiperSockets guest virtual machine network connection to also directly communicate with the following:

- Other guest virtual machines on the z/VM virtual switch

- External network hosts through the virtual switch OSA Uplink port.

The NETSTAT DEVLINKS command has been updated for this support.

Client Certificate Validation for z/VM SSL Server

The z/VM SSL server has been enhanced to accommodate client certificate validation as part of handshaking for dynamic SSL connections. This provides a means for TLS handshaking to incorporate mutual authentication of certificates before the establishment of a secure connection.

The response for the NETSTAT CONFIG PARMS command has been updated for this support.

SC24-6240-02, z/VM Version 6 Release 2

This edition includes changes to support the general availability of z/VM V6.2.

IPv6 Support for FTP Server and Client

Provide support for IPv6 in the zVM FTP client and server.

The following changes have been made as a result of this support:

- FTP command.
- NETRC DATA file.
- OPEN command.
- PASSIVE command.
- SENDPORT command.

IPv6 Support for SMTP Server and Client

Provide support for IPv6 in the zVM SMTP client and server.

GSKKYMAN

Added information for FIPS support. See Chapter 8, "SSL Certificate/Key Management and SSL Tracing Information," on page 233.

MPROUTE Enhancements and Upgrade to z/OS 1.12 Equivalency

The following information has changed to support the upgrade of MPROUTE to z/OS 1.12 equivalency.

- The NETSTAT command now includes the ROUTERADV option. See "NETSTAT Command" on page 184 and "ROUTERADV" on page 207.
- The NETSTAT GATE command and the NETSTAT CONFIG HELP command now include new output fields. See "NETSTAT Command" on page 184.

LDAP Enhancements

A new utility, LDAPCHPW (ldapchangepwd) has been added to Chapter 6, "Using the LDAP Client Utilities," on page 137.

Query OSA Address Table

This support provides TCP/IP with the capability to display the content of the OSA Address Table (OAT) via the QDIO interface for both OSA cards and VSWITCH controllers. The NETSTAT command now includes the OSAINFO option.

Support Withdrawal

The z/VM Kerberos servers (VMKERB and ADMSERV) and the z/VM Domain Name System server (NAMESRV) have been removed from TCP/IP for z/VM, Level 620. The Kerberos protocol is no longer supported. The Domain Name System (DNS) protocol is still supported using non-z/VM servers.

SC24-6240-01, z/VM Version 6 Release 1 (Updated Edition)

This edition includes changes to support product changes provided or announced after the general availability of z/VM V6.1.

z/VM SSL Server Enhancements

These enhancements improve the SSL server's ability to provide concurrent secure connectivity by increasing its overall performance and decreasing the amount of system resources it requires.

Changes have been made in Chapter 7, “Monitoring the TCP/IP Network,” on page 183.

- A new SSL option was added on the NETSTAT IDENTIFY command.
- The NETSTAT CONFIG SSL command has been updated.

Support for IBM zEnterprise System Networks

Networking support for the IBM z Unified Resource Manager enhances z/VM's virtual and real networking components to provide integrated management of these resources within an ensemble. z/VM's hypervisor provides the foundation for this management for both the intraensemble data network (IEDN) and the intranode management networks (INMN).

The NETSTAT command includes the new CONTROLLER option, which displays the current VSWITCH CONTROLLER settings. See “NETSTAT Command” on page 184 and “CONTROLLER” on page 205.

The NETSTAT CONFIG HELP output includes the new CONTROLLER option. See “HELP” on page 209.

The NETSTAT DEVLINKS command now returns two additional values in the TYPE field: IEDN (for an intra ensemble data network) and INMN (for an intra node management network). See “DEVLINKS” on page 211.

SC24-6240-00, z/VM Version 6 Release 1

This edition includes changes or additions to support the general availability of z/VM V6.1.

GSKKYMAN and GSKTRACE

Information about the GSKKYMAN and GSKTRACE commands has been moved to this document. See Chapter 8, “SSL Certificate/Key Management and SSL Tracing Information,” on page 233.

NSLOOKUP

Enhanced NSLOOKUP documentation.

Chapter 1. Introducing Computer Networks and Protocols

This chapter introduces the concepts of computer networks and an internet environment. The protocols used by TCP/IP are listed by layer and then described. Routing and addressing guidelines are also described.

Computer Networks

A computer network is a group of connected nodes that are used for data communication. A computer network configuration consists of data processing devices, software, and transmission media that are linked for information interchange.

Nodes are the functional units, located at the points of connection among the data circuits. A node, or end point, can be a host computer, a communication controller, a cluster controller, a video display terminal, or another peripheral device.

Computer networks can be local area networks (LANs), which provide direct communication among data stations on the user's local premises, or wide area networks (WANs), which provide communication services to a geographic area larger than that served by a LAN. Typically, WANs operate at a slower rate of speed than LANs.

Different types of networks provide different functions. Network configurations vary, depending on the functions required by the organization. Different organizations implement different types of networks. The technology used by these networks varies not only from organization to organization, but often varies within the same company.

Networks can differ at any or all layers. At the physical layer, networks can run over various network interfaces, such as Ethernet. Networks can also vary in the architectures they use to implement network strategies. Some of the more common architectures used today are Open Systems Interconnection (OSI), Transmission Control Protocol/Internet Protocol (TCP/IP), Systems Network Architecture (SNA), and Integrated Services Digital Network (ISDN). Networks use different protocols (rules) to communicate over the different physical interfaces available. In addition to these differences, networks can use different software packages to implement various functions.

To exchange information among these different networks, the concept of an internet emerged.

Internet Environment

An internet is a logical collection of networks supported by gateways, routers, bridges, hosts, and various layers of protocols. An internet permits different physical networks to function as a single, large, virtual network, and permits dissimilar computers to communicate with each other, regardless of their physical connections. Processes within gateways, routers, and hosts originate and receive packet information. Protocols specify a set of rules and formats required to exchange these packets of information.

© Copyright IBM Corp. 1987, 2016

Protocols are used to accomplish different tasks in TCP/IP software. To understand TCP/IP, you should be familiar with the following terms and relationships.

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 some other resource.

A ***datagram*** is a basic unit of information, consisting of one or more data packets that are passed across an internet at the transport level.

A ***gateway*** is a functional unit that connects two computer networks of different network architectures. A ***router*** is a device that connects networks at the ISO Network Layer. A router is protocol-dependent and connects only networks operating the same protocol. Routers do more than transmit data; they also select the best transmission paths and optimum sizes for packets. A ***bridge*** is a router that connects two or more networks and forwards packets among them. The operations carried out by a bridge are done at the physical layer and are transparent to TCP/IP and TCP/IP routing.

A ***host*** is a computer, connected to a network, that provides an access point to that network. A host can be a client, a server, or a client and server simultaneously. In a communication network, computers are both the sources and destinations of the packets. The ***local host*** is the computer to which a user's terminal is directly connected without the use of an internet. A ***foreign host*** is any machine on a network that can be interconnected. A ***remote host*** is any machine on a network that requires a physical link to interconnect with the network.

An ***internet address*** is a unique address identifying each node in an internet. Internet addresses are used to route packets through the network. Currently, there are two versions used for internet addressing: Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6). For more on internet addressing, see "Internet Addressing" on page 13.

Mapping relates internet addresses to physical hardware addresses in the network. For example, in IPv4, the Address Resolution Protocol (ARP) is used to map internet addresses to Ethernet physical hardware addresses. In IPv6, Internet Control Message Protocol Version 6 (ICMPv6) is used to map internet addresses to physical hardware addresses.

A ***network*** is the combination of two or more nodes and the connecting branches among them. A ***physical network*** is the hardware that makes up a network. A ***logical network*** is the abstract organization overlaid on one or more physical networks. An internet is an example of a logical network.

Packet refers to the unit or block of data of one transaction between a host and its network. A packet usually contains a network header, at least one high-level protocol header, and data blocks. Generally, the format of the data blocks does not affect how packets are handled. Packets are the exchange medium used at the internetwork layer to send and receive data through the network.

A ***port*** is 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. When the port number is combined with an internet address, a ***socket*** address results.

Protocol refers to a set of rules for achieving communication on a network.

Internet Protocol Version 4 and Internet Protocol Version 6

Internet Protocol version 4 (IPv4) is the set of protocols that most TCP/IP networks use. A new generation of protocols has been developed called Internet Protocol version 6 (IPv6). IPv4 is now approximately 20 years old and beginning to exhibit problems. The most significant issue surrounding IPv4 is the growing shortage of IPv4 addresses, which are needed by all machines attached to the Internet. IPv4 uses 32-bit addresses. In theory, 32 bits allows over 4 billion nodes, each with a globally-unique address. In practice, the interaction between routing and addressing makes it impossible to exploit more than a small fraction of that many nodes. Consequently, there is a growing concern that the continued growth of the Internet will lead to the exhaustion of IPv4 addresses early in the 21st century.

IPv6 fixes a number of problems in IPv4, such as the limited number of available IPv4 addresses. IPv6 uses 128-bit addresses, an address space large enough to last for the foreseeable future. It also adds many improvements to IPv4 in areas such as routing and network autoconfiguration. IPv6 is expected to gradually replace IPv4, with the two coexisting for a number of years during a transition period.

For general IPv6 information, see ONLamp.com: Introduction to IPv6 (www.onlamp.com/lpt/a//onlamp/2001/05/24/ipv6_tutorial.html).

Also available is a description of IPv6 in *TCP/IP Tutorial and Technical Overview*, GG24-3376, at IBM Redbooks (www.ibm.com/redbooks/).

The TCP/IP for z/VM IPv6 support improves the guest LAN support for the OSA-Express® simulation in QDIO mode and for HiperSockets simulation. Virtual machines (z/VM and other guest operating systems) in the guest LAN environment are able to define and to use simulated devices that support both the IPv4 and IPv6 protocols. The IP Assist Simulation for QDIO-based and HiperSockets-base network adapters has been updated to allow virtual machines to detect that their OSA-Express devices support both IPv4 and IPv6 and interact with these devices according to the IP Assists architecture.

The current IPv6 support in TCP/IP for z/VM is at the network (IP) layer. (For more information about TCP/IP functions grouped by layer, see "TCP/IP Protocols and Functions" on page 5.) The TCP/IP for z/VM stack allows routing IPv6 traffic in the guest LAN environment. Operationally, a single TCP/IP for z/VM stack provides support for static routing of IPv6 traffic and provides the existing IPv4 support. The TCP/IP for z/VM stack does not support IPv6 security or IPv6 in the upper application layers, such as SMTP. An exception to upper application layer support is Telnet: a telnet client that supports IPv6 can connect to the telnet server in the TCP/IP for z/VM stack.

Additionally, programming interfaces for the sockets library now support IPv6. The support includes updates to the sockets interfaces for the TCP/IP for z/VM stack, the Byte File System, and Language Environment®. For more information on the programming interfaces, see:

- *z/VM: TCP/IP Programmer's Reference*
- *z/VM: OpenExtensions Callable Services Reference*
- *XL C/C++ for z/VM: Runtime Library Reference*

TCP/IP for z/VM currently supports the following IPv6-related RFCs:

- *RFC 2460, Internet Protocol, Version 6 (IPv6), Specification*

The IPv6 specification defines the basic IPv6 header and the IPv6 extension headers and options. The specification also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols.

- *RFC 3513, Internet Protocol Version 6 (IPv6) Addressing Architecture*

  The IPv6 Addressing Architecture specification defines the addressing architecture of the IPv6 protocol. It includes a detailed description of the currently defined address formats for IPv6.

- *RFC 2461, Neighbor Discovery for IP Version 6 (IPv6)*

  The neighbor discovery specification defines how to do address resolution for neighboring nodes (similar to ARP for IPv4), as well as how to locate neighboring routers (this is called *router discovery*). For IPv4, the OSA-Express adapter provides the offload function of maintaining the ARP cache. For IPv6, the address resolution function of neighbor discovery is not offloaded to the OSA-Express adapter and is implemented in the TCP/IP for z/VM stack.

  The TCP/IP for z/VM stack implements both the host and router parts of the neighbor discovery protocol. When configured as a router, router advertisements can be sent to provide autoconfiguration information for other hosts-prefixes, parameters and default routes.

  Restriction: The TCP/IP for z/VM stack cannot be configured as a tunnel endpoint for tunneling IPv6 traffic over IPv4 networks.

- *RFC 2462, IPv6 Stateless Address Autoconfiguration*

  The specification defines the steps a host takes in deciding how to autoconfigure its interfaces in IPv6.

- *RFC 2710, Multicast Listener Discovery (MLD) for IPv6*

  The specification defines the protocol used by an IPv6 router to discover the presence of multicast listeners (that is, nodes wishing to receive multicast packets) on its directly-attached links, and to discover specifically which multicast addresses are of interest to those neighboring nodes.

  Restriction: The TCP/IP for z/VM stack participates in multicast listener discovery performing the host function of registering multicast addresses needed for neighbor discovery. However, the TCP/IP for z/VM stack is not a router of IPv6 multicast traffic (a multicast router).

- *RFC 2463, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

  The ICMPv6 specification defines the packet formats and processing rules for ICMP for IPv6.

- *RFC 3484, Default Address Selection for Internet Protocol version 6 (IPv6)*

  The specification describes two algorithms; one for source address selection and one for destination address selection.

  Restriction: The TCP/IP for z/VM stack uses the source address selection algorithm only.

- *RFC 5095, Deprecation of Type 0 Routing Headers in IPv6*

  The specification describes the functionality provided by IPv6's Type 0 Routing Header, which can be exploited in order to achieve traffic amplification over a remote path for the purposes of generating denial-of-service traffic. RFC5095 updates the IPv6 specification to deprecate the use of IPv6 Type 0 Routing Headers, in light of this security concern.

- *RFC 5722, Handling of Overlapping IPv6 Fragments*

The specification discusses an attack that can be used to bypass IPv6 firewalls using overlapping fragments. It recommends disallowing overlapping fragments in order to prevent this attack.

- *RFC 6946, Processing of IPv6 "Atomic" Fragments*

  The specification discusses the generation of the atomic fragments and the corresponding security implications. It introduces how to process atomic fragments independently of any other fragments to completely eliminating the attack vector.

- *RFC 6980, Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery*

  The specification analyzes the security implications of employing IPv6 fragmentation with Neighbor Discovery (ND) messages. It updates RFC 4861 such that use of the IPv6 Fragmentation Header is forbidden in all Neighbor Discovery messages, thus allowing for simple and effective countermeasures for Neighbor Discovery attacks. It also discusses the security implications of using IPv6 fragmentation with SEcure Neighbor Discovery (SEND) and formally updates RFC 3971 to provide advice regarding how the aforementioned security implications can be mitigated.

  Restriction: The TCP/IP for z/VM stack does not support SEcure Neighbor Discovery; the specification about forbidding IPv6 Fragmentation with IPv6 SEcure Neighbor Discovery is ignored.

Copies of these RFCs are available at IP Version 6 Working Group (ipv6) - Charter (datatracker.ietf.org/ipv6/charter/)

TCP/IP Protocols and Functions

This section categorizes the TCP/IP protocols and functions by their functional group link (physical) layer, network layer, transport layer, and application layer). Table 2 shows the functional groups and their related protocols and functions.

Table 2. Functional Groups

Group	Protocols and Functions	Location
Link (physical) layer	Ethernet Others	"Link Protocols" on page 6
Network Layer	Internet Protocol (IP) Internet Control Message Protocol (ICMP) Address Resolution Protocol (ARP) Internet Group Management Protocol (IGMP) Internet Protocol version 6 (IPv6) Internet Control Message Protocol Version 6 (ICMPv6) Neighbor Discovery Stateless Address Autoconfiguration Multicast Listener Discovery	"Network Protocols" on page 7
Transport Layer	Transmission Control Protocol (TCP) User Datagram Protocol (UDP)	"Transport Protocols" on page 9

Table 2. Functional Groups (continued)

Group	Protocols and Functions	Location
Application Layer	Telnet File Transfer Protocol (FTP) Internet Message Access Protocol (IMAP) Simple Mail Transfer Protocol (SMTP) Domain Name System (DNS) Simple Network Management Protocol (SNMP) Remote Printing (LPR and LPD) MPRoute X Window System X Toolkit GDDMXD Remote Procedure Call (RPC) Network File System (NFS) Remote Execution Protocol (REXEC) Socket Interfaces Secure Socket Layer (SSL)	"Applications and Protocols" on page 9

Figure 1 shows the relationship of these protocols and functions within the TCP/IP layered architecture for VM.

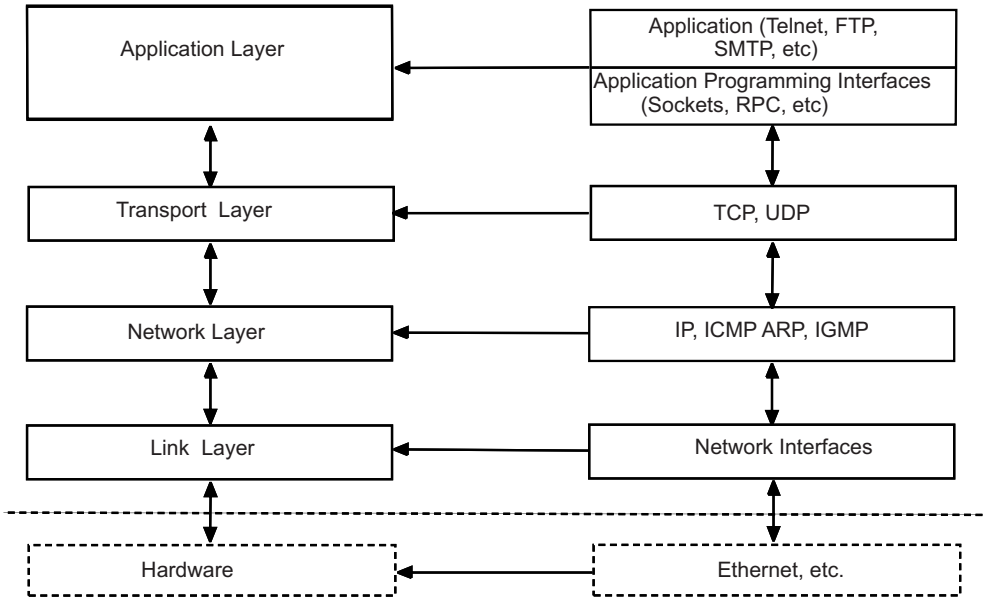


Figure 1. The TCP/IP Layered Architecture

Link Protocols

Various network protocols compose the network layer available in TCP/IP. Network protocols define how data is transported over a physical network. These network protocols are not defined by TCP/IP. After a TCP/IP packet is created, the network protocol adds a transport-dependent network header before the packet is sent out on the network.

Network Protocols

Protocols in the internetwork layer provide connection services for TCP/IP. These protocols connect physical networks and transport protocols. This section describes the internetwork protocols in TCP/IP.

For more information about TCP/IP in general, see RFCs 1118, 1180, 1206, 1207, and 1208. For a list of other related RFCs, see Appendix G, "Related Protocol Specifications," on page 497.

Internet Protocol (IP)

The Internet Protocol (IP) provides the interface from the transport layer (host-to-host, TCP, or UDP) protocols to the physical-level protocols. IP is the basic transport mechanism for routing IP packets to the next gateway, router, or destination host.

IP provides the means to transmit blocks of data (or packets of bits) from sources to destinations. Sources and destinations are hosts identified by fixed-length internet addresses. Outgoing packets automatically have an IP header prefixed to them, and incoming packets have their IP header removed before being sent to the higher-level protocols. This protocol provides for the universal addressing of hosts in an internet network.

IP does not ensure a reliable communication, because it does not require acknowledgments from the sending host, receiving host, or intermediate hosts. IP does not provide error control for data; it provides only a header checksum. IP treats each packet as an independent entity unrelated to any other packet. IP does not perform retransmissions or flow control. A higher-level protocol that uses IP must implement its own reliability procedures.

For more information about IP, see RFC 791.

Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) passes control messages between hosts, gateways, and routers. For example, ICMP messages can be sent in any of the following situations:

- When a host checks to see if another host is available (with a PING command)
- When a packet cannot reach its destination
- When a gateway or router can direct a host to send traffic on a shorter route
- When a host requests a netmask or a time stamp
- When a gateway or router does not have the buffering capacity to forward a packet

ICMP provides feedback about problems in the communication environment; it does not make IP reliable. ICMP does not guarantee that an IP packet is delivered reliably or that an ICMP message is returned to the source host when an IP packet is not delivered or is incorrectly delivered.

For more information about ICMP, see RFC 792.

Address Resolution Protocol (ARP)

The Address Resolution Protocol (ARP) maps internet addresses to hardware addresses. TCP/IP uses ARP to collect and distribute the information for mapping tables.

ARP is not directly available to users or applications. When an application sends an internet packet, IP requests the appropriate address mapping. If the mapping is not in the mapping table, an ARP broadcast packet is sent to all the hosts on the network requesting the physical hardware address for the host.

For more information about ARP, see RFC 826.

Internet Group Management Protocol (IGMP)

The Internet Group Management Protocol (IGMP) is used to communicate multicast group information. It lets all systems on a physical network know which hosts belong to which multicast groups. Multicast routers use IGMP messages to determine which multicast datagrams to forward onto which interfaces.

There are two types of IGMP messages — reports and queries. IGMP report messages are sent out to notify others when a multicast group has been joined, and to respond to an IGMP query that was received. IGMP query messages are sent out by multicast routers to see which multicast groups still have members.

For more information about IGMP, see RFC 1112.

Internet Control Message Protocol Version 6 (ICMPv6)

In addition to the functions carried out by ICMP for IPv4, ICMPv6 conveys multicast group membership information, a function previously performed by IGMP, and address resolution, a function previously performed by ARP.

For more information about ICMPv6, see RFC 2463.

Neighbor Discovery

Neighbor discovery is an ICMPv6 function that enables a node to identify other hosts and routers on its links. The node needs to know at least one router in order to forward packets to a target node not on its local link. Neighbor discovery also allows a router to tell a node to use a more appropriate router if that node has initially made an incorrect choice.

For more information on neighbor discovery, see RFC 2461.

Stateless Address Autoconfiguration

Although the 128-bit address field of IPv6 solves a number of problems inherent in IPv4, the size of the IPv6 address space represents a potential problem to the TCP/IP administrator. Due to this potential problem, IPv6 has the capability, through stateless address autoconfiguration, to assign an address to an interface automatically at initialization time, with minimal or no action by the TCP/IP administrator.

For more information on stateless address autoconfiguration, see RFC 2462.

Multicast Listener Discovery

Multicast listener discovery (MLD) is a process used by a router to discover the members of a multicast group. MLD is a subset of ICMPv6 and provides the equivalent function in IGMP for IPv4. Through MLD, information is provided by the router to whichever multicast routing protocol is being used, so that multicast packets are correctly delivered to all links where there are nodes listening for the appropriate multicast address.

For more information on multicast listener discovery, see RFC 2710.

Transport Protocols

The transport layer of TCP/IP consists of transport protocols, which allow communication between application programs. This section describes the transport protocols in TCP/IP.

Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) provides a reliable vehicle for delivering packets between hosts on an internet. TCP takes a stream of data, breaks it into datagrams, sends each one individually using IP, and reassembles the datagrams at the destination node. If any datagrams are lost or damaged during transmission, TCP detects this and resends the missing datagrams. The received data stream is a reliable copy of the transmitted data stream.

For more information about TCP, see RFC 793.

User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) provides an unreliable mode of communication between source and destination hosts. UDP is a datagram-level protocol built directly on the IP layer. UDP is used for application-to-application programs between TCP/IP hosts.

Like IP, UDP does not offer a guarantee of datagram delivery or duplication protection. UDP does provide checksums for both the header and data portions of a datagram. However, applications that require reliable delivery of streams of data should use TCP.

For more information about UDP, see RFC 768.

Applications and Protocols

Applications are provided with TCP/IP that allow users to use network services. These applications are included in the application layer of TCP/IP. The application layer is built on the services of the transport layer. This section describes the applications, functions, and protocols in TCP/IP.

Telnet Protocol

The Telnet Protocol provides a standard method to interface terminal devices and terminal-oriented processes with each other. Telnet is built on the services of TCP in the transport layer. Telnet provides duplex communication and sends data either as ASCII characters or as binary data.

Telnet is commonly used to establish a logon session on a foreign host. Telnet can also be used for terminal-to-terminal communication and interprocess communication.

For more information about the Telnet Protocol, see RFCs 854, 856, 857, 885, and 1091.

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) allows you to transfer data between local and foreign hosts or between two foreign hosts. FTP is built on the services of TCP in the transport layer. FTP transfers files as either ASCII characters or as binary data. ASCII characters are used to transfer files that contain only text characters.

FTP provides functions, such as listing remote directories, changing the current remote directory, creating and removing remote directories, and transferring one or more files in a single request. Security is handled by passing user and account passwords to the foreign hosts.

For more information about FTP, see RFC 959.

Internet Message Access Protocol (IMAP)

The Internet Message Access Protocol (IMAP) is a mail protocol with both client (sender) and server (receiver) functions.

IMAP provides the processing that allows a client to access electronic mail that is kept in an IMAP Mailstore server. It permits a "client" email program to access remote message folders called "mailboxes", as if they were local.

For more information about IMAP, see RFC 2060.

Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP) is an electronic mail protocol with both client (sender) and server (receiver) functions.

SMTP is implemented with the CMS NOTE and CMS SENDFILE EXECs in a VM environment. You do not interface directly with SMTP. Instead, electronic mail software is used to create mail, which in turn uses SMTP to send the mail to its destination.

For more information about SMTP, see RFCs 821, 822, 974, 1413, and 1440.

Domain Name System (DNS)

The Domain Name System (DNS) uses a hierarchical-naming system for naming hosts. Each host name is composed of domain labels separated by periods. Local network administrators have the authority to name local domains within an internet. Each label represents an increasingly higher domain level within an internet. The fully qualified domain name of a host connected to one of the larger internets generally has one or more subdomains.

For example:

host.subdomain.subdomain.rootdomain
or
host.subdomain.rootdomain

For more information about the Domain Name System, see RFCs 1034 and 1035.

Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) provides a means for managing an internet environment. SNMP allows network management by elements, such as gateways, routers, and hosts. Network elements act as servers and contain management agents, which perform the management functions requested. Network management stations act as clients; they run the management applications, which monitor the network. SNMP provides a means of communicating between these elements and stations to send and receive information about network resources.

For more information about Network Management, see RFCs 1155, 1157, 1187, and 1213.

Remote Printing (LPR)

TCP/IP provides client support for remote printing. This application allows you to spool files remotely to a line printer daemon (LPD). The line printer client (LPR) sends the file to be printed to a specified print server host and to a specified printer.

For more information about LPR and LPD, see RFC 1179.

MPRoute

MPRoute uses the Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) protocol to create and maintain network routing tables dynamically. RIP and OSPF arrange to have gateways and routers periodically broadcast their routing tables to neighbors. Using this information, an MPRoute server can update a host's routing tables. For example, MPRoute determines if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists.

For more information about RIP, see RFCs 1058 and 1723. For more information about OSPF, see RFC 1583.

X Window System

The X Window System Protocol is designed to support network transparent windowing and graphics. TCP/IP for z/VM provides client support for the X Window System application program interface.

For more information about the X Window System Protocol, see RFC 1013.

X Toolkit

X toolkit is a collection of basic C language routines for developing a variety of application environments.

Intrinsics: Intrinsics are a collection of C language routines for building and using widgets.

Widgets: Widgets are a set of routines that create a graphic interface.

- Athena Widgets

  The widget set developed by MIT's Project Athena is generally known as the Athena Widget set. It enables application programmers to include standard graphic elements in their applications.
- OSF/Motif

  OSF/Motif** is an X Window System toolkit defined by Open Software Foundation, Inc. (OSF**), which enables the application programmer to include standard graphic elements that have a three-dimensional appearance. Performance of the graphic elements is increased with gadgets and windowless widgets.

GDDMXD

GDDMXD is an interface that allows graphics from the IBM Graphical Data Display Manager/VM to be displayed on workstations that support the X Window System.

When GDDMXD is installed and activated, the data stream created by GDDM is translated to the X Window System protocol and transmitted by TCP/IP to the X Window System server for the display. If GDDMXD is installed and not activated, or has been made inactive, GDDM transmits data to its supported display device as if GDDMXD was not present.

Remote Procedure Call (RPC)

The Remote Procedure Call Protocol (RPC) is a programming interface that calls subroutines to be executed on a foreign host. RPCs are high-level program calls, which can be used in place of the lower-level calls that are based on sockets.

For more information about RPC, see RFC 1057.

Network File System (NFS)

The Network File System (NFS) allows you to manipulate files on different TCP/IP hosts as if they reside on your host. NFS is based on the NFS protocol, and uses the Remote Procedure Call (RPC) protocol to communicate between the client and the server. The files to be accessed reside on the server host and are made available to the user on the client host.

The Network File System supports the hierarchical file structure used by the UNIX operating system. The directory and subdirectory structure can be different for individual client systems.

For more information about NFS, see RFC 1094 and 1813.

Remote Execution Protocol (REXEC)

The Remote Execution Protocol (REXEC) allows you to execute a command on a foreign host and receive the results on the local host. Remote Execution Protocol provides automatic logon and user authentication, depending on the parameters set by the user.

Socket Interfaces

Socket interfaces allow you to write your own applications to supplement those supplied by TCP/IP. Most of these additional applications communicate with either TCP or UDP. Some applications are written to communicate directly with IP. To write applications that use the socket interfaces of TCP/IP for z/VM, you must be able to compile and link the programs.

Sockets are duplex, which means that data can be transmitted and received simultaneously. Sockets allow you to send to, and receive from, the socket as if you are writing to and reading from any other network device. For more information on sockets, see *z/VM: TCP/IP Programmer's Reference*.

Secure Socket Layer (SSL)

The Secure Socket Layer (SSL) protocol provides privacy between two communicating applications — a client and a server. In SSL, a server is always authenticated and must provide a certificate to prove its identity. In addition to authentication, both the client and the server participate in a handshake protocol that produces the cryptographic parameters for the session.

The processing required for SSL is provided by a TCP/IP security server. An installation identifies the ports that are secure and specifies the certificates to be used. Any VM TCP/IP server listening on a secure port can participate in an SSL session with any external client that supports SSL. The SSL session consists of two connections — the connection from the remote client to the security server and the connection from the security server to the real (application) server.

For more information about configuring the SSL Server, see *z/VM: TCP/IP Planning and Customization*. For information about managing SSL certificates and obtaining SSL diagnostic information, see Chapter 8, "SSL Certificate/Key Management and SSL Tracing Information," on page 233.

Routing

The routing functions in an internet are performed at the network layer. Routing is the process of deciding where to send a packet based on its destination address. Two kinds of routing are involved in communication within an internet: direct and indirect.

Direct routing is used when the source and destination nodes are on the same logical network within an internet. The source node maps the destination internet address into a hardware address and sends packets to the destination node through this address. This mapping is normally performed through a translation table. If a match cannot be found for a destination internet address, ARP in IPV4 or neighbor discovery in IPv6 is invoked to determine this address.

Indirect routing is used when the source and destination nodes are on different networks within an internet. The source node sends packets to a gateway or router on the same network using direct routing. From there, the packets are forwarded through intermediate gateways or routers, as required, until they arrive at the destination network. Direct routing is then used to forward the packets to the destination host on that network. Each gateway, router, and host in an internet has a routing table that defines the address of the next gateway or router to other networks (as well as other nodes on other networks) in an internet.

Internet Addressing

Each internet host is assigned at least one unique internet address. This address is used by the IP and other higher-level protocols. When gateway hosts are used, more than one address may be required. Each interface to an internet is assigned its own unique address. Internet addresses are used to route packets through the network.

Internet addressing differs between IPv4 and IPv6. The next topic covers IPv4 addressing, followed by a topic on IPv6 addressing.

IPv4 Addressing

Addresses within an internet consist of a network number and a local address. The unique network number is assigned to each network when it connects to another internet. If a local network is not going to connect to other internets, any convenient network number is assigned.

Hosts that exchange packets on the same physical network should have the same network number. Hosts on different physical networks might also have the same network number. If hosts have the same network number, part of the local address is used as a subnetwork number. All host interfaces to the same physical network are given the same subnetwork number.

An internet can provide standards for assigning addresses to networks, broadcasts, and subnetworks. Examples of these standard formats are described in the following sections.

Network Address Format

A standard internet address uses a two-part, 32-bit address field. The first part of the address field contains the network address; the second part contains the local address. The four different types of address fields are classified as A, B, C, or D, depending on the bit allocation.

Figure 2 represents a class A address. A class A address has a 7-bit network number and a 24-bit local address. The highest order bit is set to 0.

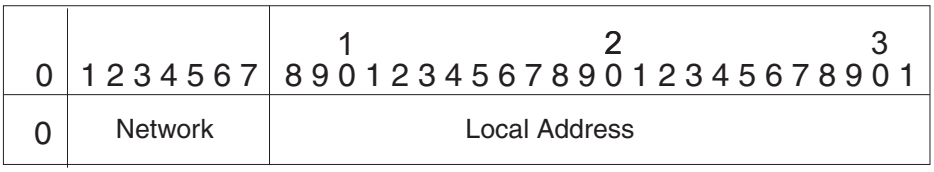


Figure 2. Class A Address

Figure 3 represents a class B address. A class B address has a 14-bit network number and a 16-bit local address with the highest order bits set to 10.

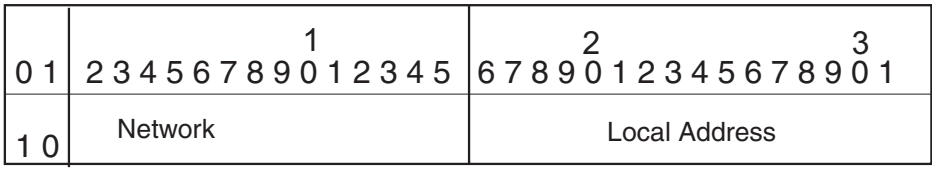


Figure 3. Class B Address

Figure 4 represents a class C address. A class C address has a 21-bit network number and an 8-bit local address with the three highest order bits set to 110.

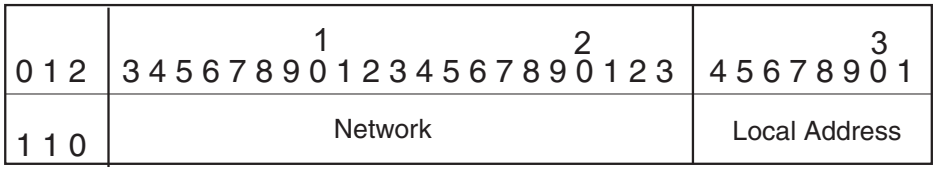


Figure 4. Class C Address

Figure 5 represents a class D address. A class D network is a multicast address that is sent to selected hosts on the network. The four highest order bits are set to 1110.

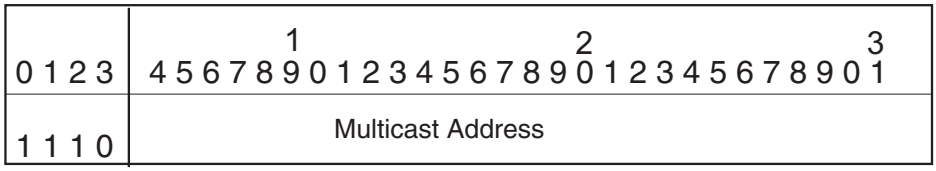


Figure 5. Class D Address

Note: Class D addresses are not supported in TCP/IP for z/VM.

A commonly used notation for internet host addresses is the dotted-decimal, which divides the 32-bit address into four 8-bit fields. The value of each field is specified as a decimal number, and the fields are separated by periods (for example, 010.002.000.052 or 10.2.0.52).

Address examples in this book use dotted-decimal notation in the following forms:
Class A

nnn.lll.lll.lll

Class B
nnn.nnn.lll.lll
Class C
nnn.nnn.nnn.lll

where:
nnn Represents part or all of a network number.
lll Represents part or all of a local address.

Broadcast Address Format

TCP/IP uses IP broadcasting to send datagrams to all the TCP/IP hosts on a network or subnetwork. A datagram sent to the broadcast address is received by all the hosts on the network and processed as if the datagram was sent directly to the host's IP address. The IP broadcast address is formed by setting all the host bits to ones.

For more information about IP broadcasting, see RFCs 919 and 922.

Multicast Address Format

TCP/IP uses IP multicasting to send datagrams to all the TCP/IP hosts on a network or subnetwork. The multicast datagrams are only received by those TCP/IP hosts that have signed up to listen for the particular IP multicast address (joined the multicast group). If a TCP/IP host has not joined the multicast group, then the datagram is discarded.

For more information on IP multicasting, see RFC 1112.

Subnetwork Address Format

The subnetwork capability of TCP/IP divides a single network into multiple logical networks (subnets). For example, an organization can have a single internet network address that is known to users outside the organization, yet configure its internal network into different departmental subnets. Subnetwork addresses enhance local routing capabilities, while reducing the number of network numbers required. For a subnet, the local address part of an internet address is divided into a subnet number and a host number, for example:

network_number subnet_number host_number

where:
network_number
Is the network portion of the internet address.
subnet_number
Is a field of a constant width for a given network.
host_number
Is a field that is at least 1-bit wide.

If the width of the *subnet_number* field is 0, the network is not organized into subnets, and addressing to the network is done with an internet network address (*network_number*).

Figure 6 on page 16 represents a class B address with a 6-bit wide subnet field.

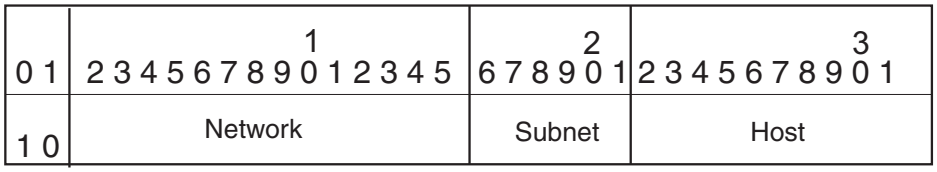


Figure 6. Class B Address with Subnet

The bits that identify the subnet are specified by a bit mask. A bit mask is a pattern of characters used to assign subnet addresses. The subnet bits are not required to be adjacent in the address. However, the subnet bits generally are contiguous and are the most significant bits of the local address.

For more information about subnetwork addresses, see RFC 950.

IPv6 Addressing

One problem that IPv6 solves is the limited number of addresses available in IPv4. IPv6 uses a 128-bit address space, which has no practical limit on global addressibility and provides 340 282 366 920 938 463 463 374 607 431 768 211 456 addresses. Currently, this is enough addresses so that every person can have a single IPv6 network with as many as 18 000 000 000 000 000 000 nodes on it, and still the address space would be almost completely unused.

There are three conventional forms for representing IPv6 addresses as text strings:

- The preferred form is x:x:x:x:x:x:x:x, where the x's are the hexadecimal values of the eight 16-bit pieces of the address.

  Examples:

  ```
  FE80:0000:0000:0000:0001:0800:23e7:f5db

  1080:0:0:0:8:800:200C:417A
  ```

  It is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in the following bullet).
- Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. The use of :: indicates multiple groups of 16 bits of zeros. The :: can only appear once in an address. The :: can also be used to compress both leading and trailing zeros in an address.

  Examples: The following are preferred form addresses:

  ```
  1080:0:0:0:8:800:200C:417A  a unicast address
  FF01:0:0:0:0:0:0:101        a multicast address
  0:0:0:0:0:0:0:1             the loopback address
  0:0:0:0:0:0:0:0             the unspecified addresses
  ```

  The corresponding compressed forms are:

  ```
  1080::8:800:200C:417A       a unicast address
  FF01::101                   a multicast address
  ::1                         the loopback address
  ::                          the unspecified addresses
  ```
- An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the x's are the hexadecimal values of the 6 high-order 16-bit pieces of the address, and the d's are the decimal values of the 4 low-order 8-bit pieces of the address (standard IPv4 representation). This form is used for IPv4-compatible IPv6

addresses and IPv4-mapped IPv6 addresses. These types of addresses are used to hold embedded IPv4 addresses in order to carry IPv6 packets over the IPv4 routing infrastructure.

Examples:

```
0:0:0:0:0:0:13.1.68.3
0:0:0:0:0:FFFF:129.144.52.38
```

The same addresses in compressed form are:

```
::13.1.68.3
::FFFF:129.144.52.38
```

Hierarchical Addressing and Routing Infrastructure

As important as the expanded address space is the use of hierarchical address formats. The IPv4 addressing hierarchy includes network, subnet, and host components in an IPv4 address. IPv6, with its 128-bit addresses, provides globally unique and hierarchical addressing based on prefixes rather than address classes, which keeps routing tables small and backbone routing efficient.

The general format is as follows:

Table 3. IPv6 Address Format

global routing prefix	subnet ID	interface ID
n bits	m bits	128-(n+m) bits

The global routing prefix is a value (typically hierarchically structured) assigned to a site; the subnet ID is an identifier of a link within the site; and the interface ID is a unique identifier for a network device on a given link (usually automatically assigned).

When configured for unicast routing, the TCP/IP for z/VM stack may also be configured to provide autoconfiguration information for other hosts-prefixes, parameters, and default routes, as specified in the *RFC 2461, Neighbor Discovery for IP Version 6 (IPv6)*.

Textual representation of IPv6 prefixes

The text representation of IPv6 address prefixes is similar to the way IPv4 address prefixes are written in Classless Inter-Domain Routing (CIDR) notation. An IPv6 address prefix is represented by the notation:

```
ipv6-address/prefix-length
```

Where:

ipv6-address
Is an IPv6 address in any of the notations listed above.

prefix-length
Is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

Example: The following are legal representations of the 60-bit prefix 12AB00000000CD3 (hexadecimal):

```
12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60
```

When writing both a node address and a prefix of that node address (for example, the node's subnet prefix), the two can be combined as follows:

```
node address:      12AB:0:0:CD30:123:4567:89AB:CDEF

its subnet number: 12AB:0:0:CD30::/60

combination:       12AB:0:0:CD30:123:4567:89AB:CDEF/60
```

Types and Categories of IPv6 Addresses

The type of a IPv6 address is identified by the high-order bits of the address, as follows:

Table 4. Types of IPv6 Addresses

Address type	Binary prefix	IPv6 notation
Unspecified	00 . . . 0 (128 bits)	::/128
Loopback	00 . . . 1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-local unicast	1111111010	FE80::/10
Site-local unicast	1111111011	FEC0::/10
Global unicast	(everything else)	

Three categories of IP addresses are supported in IPv6:

Unicast
: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. It can be link-local scope, site-local scope, or global scope.

Multicast
: An identifier for a group of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

Anycast
: An identifier for a group of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to the closest member of a group, according to the routing protocols' measure of distance.

  Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses. Anycast is described as a cross between unicast and multicast. Like multicast, multiple nodes may be listening on an anycast address. Like unicast, a packet sent to an anycast address will be delivered to one (and only one) of those nodes. The exact node to which it is delivered is based on the IP routing tables in the network.

There are no broadcast addresses in IPv6. Multicast addresses have superseded this function.

Unicast IPv6 Addresses: IPv6 unicast addresses can be aggregated with prefixes of arbitrary bit-length similar to IPv4 addresses under Classless Interdomain Routing (CIDR).

A unicast address has the following format:

Table 5. Unicast Address Format

n bits	128-n bits
network prefix	interface ID

There are several types of unicast addresses in IPv6: global unicast, site-local unicast, and link-local unicast. There are also some special-purpose subtypes of global unicast, such as IPv6 addresses with embedded IPv4 addresses. Additional address types or subtypes can be defined in the future.

Global Unicast Addresses: The general format for IPv6 global unicast addresses is:

Table 6. Global Unicast Address Format

nbits	m bits	128-n-m bits
global routing prefix	subnet ID	interface ID

The global routing prefix is a (typically hierarchically-structured) value assigned to a site (a cluster of subnets/links). The subnet ID is an identifier of a link within the site. The interface ID is used to identify an interface on a link; interface IDs are required to be unique within a subnet prefix.

All global unicast addresses other than those that start with B'000' have a 64-bit interface ID field (that is, n + m = 64). Global unicast addresses that start with B'000' have no such constraint on the size or structure of the interface ID field.

Examples of global unicast addresses that start with B'000' are IPv6 address with embedded IPv4 addresses. These include IPv4-mapped IPv6 addresses and IPv4-compatible IPv6 addresses.

Local Use Address: There are two types of local-use unicast addresses defined: link-local and site-local. The link-local address is for use on a single link and the site-local address is for use in a single site.

Link-local Addresses: Link-local addresses have the following format:

Table 7. Link-local address format

10 bits	54 bits	64 bits
1111111010	0	interface ID

A link-local address is required on each physical interface. Link-local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or in the absence of routers. It also may be used to communicate with other nodes on the same link. A link-local address is automatically assigned.

Routers will not forward any packets with link-local source or destination addresses to other links.

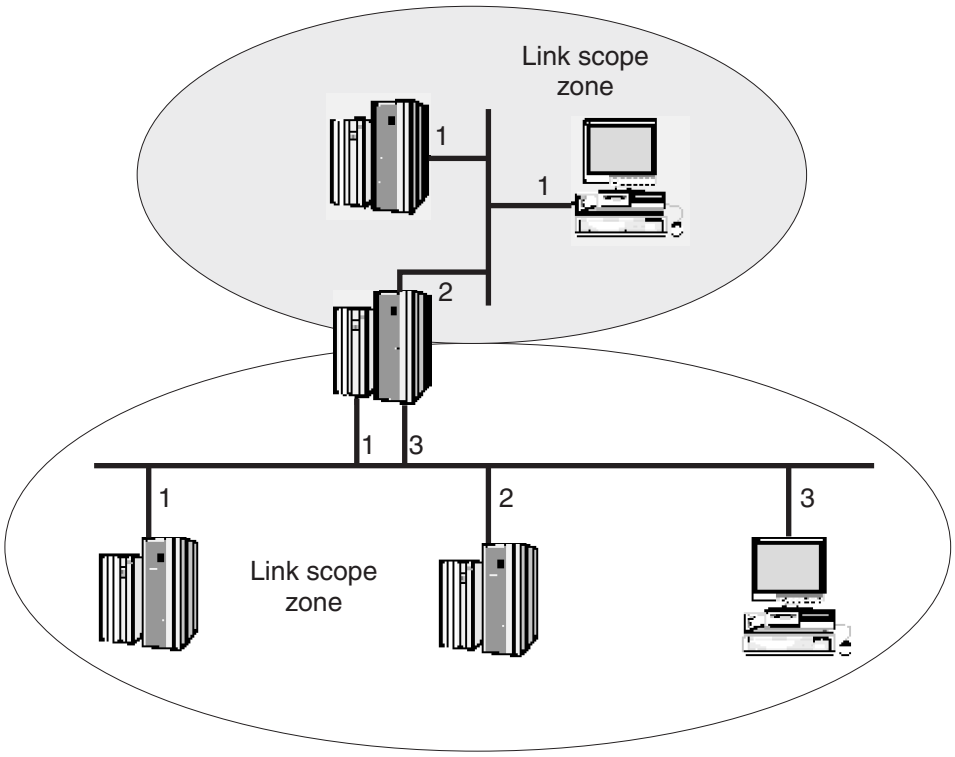


Figure 7. Link-local scope zones

Figure 7 depicts two separate link-local scope zones. More than one interface may be connected to the same link for fault tolerance or extra bandwidth. Some nodes may allow the same link-local zone index to be assigned to each interface connected to the same physical link, while others may assign a unique link-local zone index to each interface even when more than one interface is connected to the same physical link. The TCP/IP for z/VM server assigns a unique link-local address to each physical interface.

Site-local Addresses: Site-local addresses have the following format:

Table 8. Site-local address format

10 bits	38 bits	16 bits	64 bits
1111111011	0	subnet ID	interface ID

Site-local addresses are designed to be used for addressing inside of a site without the need for a global prefix. A site-local address cannot be reached from another site. A site-local address is not automatically assigned to a node. It must be assigned using automatic or manual configuration.

Routers will not forward any packets with site-local source or destination addresses outside of the site.

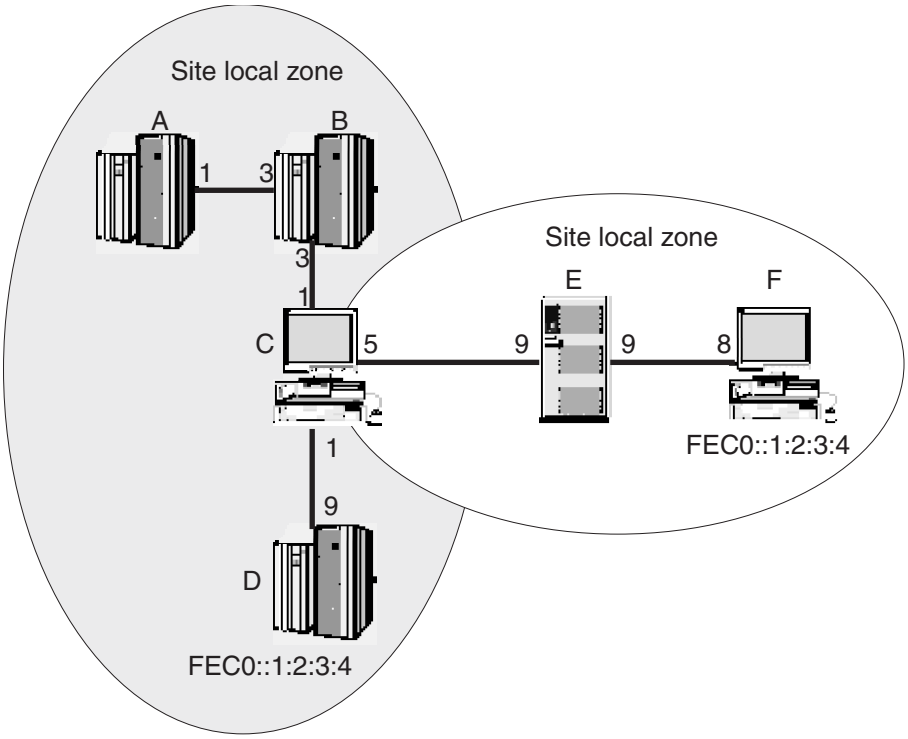


Figure 8. Site-local scope zones

Nodes connected to the same site-local scope zone may communicate with each other using site-local addresses. However, nodes which are not connected to the same site-local scope zone may not communicate using site-local addresses but must instead use global addresses.

Figure 8 depicts two site-local scope zones. In this configuration, node A can communicate with node D using site-local addresses since they are both within the same site-local scope zone. However, node A cannot communicate with node F using site-local addresses because the two nodes are not connected to the same site-local scope zone. Instead, node A must use global addresses when communicating with node F. Since node C is connected to both site-local scope zones, it may use the appropriate site-local address when communicating with both node A and node F.

The TCP/IP for z/VM server supports connecting to a single site-local scope zone and cannot be connected to two or more site-local scope zones at the same time. For example, the TCP/IP for z/VM server could be either node A or node F in Figure 8, as both are connected to only a single site-local scope zone, but could not be node C, as node C is connected to two site-local scope zones.

Loopback Address: The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It cannot be assigned to any physical interface. It may be thought of as a link-local unicast address assigned to a virtual interface (typically called the loopback interface) that allows local applications to send messages to each other.

The loopback address cannot be used as the source address in IPv6 packets that are sent outside of a node. An IPv6 packet with a destination address of loopback cannot be sent outside of a node and be forwarded by an IPv6 router. A packet received on an interface with destination address of loopback will be dropped.

Unspecified Address: The address 0:0:0:0:0:0:0:0 is called the unspecified address. It will not be assigned to any node. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

The unspecified address cannot be used as the destination address of IPv6 packets or in IPv6 routing headers. An IPv6 packet with a source address of unspecified cannot be forwarded by an IPv6 router.

IPv4-mapped IPv6 Addresses: These addresses hold an embedded global IPv4 address. They are used to represent the addresses of IPv4 nodes as IPv6 addresses to applications that are enabled for IPv6 and are using AF_INET6 sockets. This allows IPv6 enabled applications always to deal with IP addresses in IPv6 format regardless of whether the TCP/IP communications are occurring over IPv4 or IPv6 networks. The dual-mode TCP/IP stack performs the transformation of the IPv4-mapped addresses to and from native IPv4 format. IPv4-mapped addresses have the following format:

Table 9. IPv4-mapped IPv6 address

80 bits	16	32 bits
0000...0000	FFFF	IPv4 address

Examples:

- In IPv6-IPv4 decimal form:
  `::FFFF:129.144.52.38`
- In IPv6-compressed form
  `::FFFF:8190:3426`

IPv4-compatible IPv6 Addresses: These addresses hold an embedded global IPv4 address. They are used dynamically to tunnel IPv6 packets over IPv4 networks. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses which hold an IPv4 address in the low-order 32-bits. IPv4-compatible IPv6 addresses have the following format:

Table 10. IPv4-compatible IPv6 address

80 bits	16	32 bits
0000...0000	0000	IPv4 address

Examples:

- In IPv6-IPv4 decimal form
  `::129.144.52.38`
- In IPv6-compressed form
  `::8190:3426`

Multicast IPv6 Addresses: An IPv6 multicast address is an identifier for a group of interfaces (typically on different nodes). It is identified with a prefix of 11111111 or FF in hexadecimal notation. It provides a way of sending packets to multiple destinations. An interface may belong to any number of multicast groups.

Multicast address format: Binary 11111111 at the start of the address identifies the address as being a multicast address. Multicast addresses have the following format:

Table 11. Multicast address format

8	4	4	112 bits
11111111	flags	scope	group ID

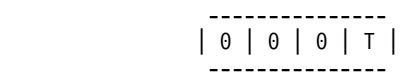


Figure 9. Flags in multicast address

- The 3 high-order flags are reserved, and must be initialized to 0.
- T = 0 indicates a permanently-assigned (well-known) multicast address, assigned by the Internet Assigned Number Authority (IANA).
- T = 1 indicates a non-permanently assigned (transient) multicast address.

Scope is a 4-bit multicast scope value used to limit the scope of the multicast group. Group ID identifies the multicast group, either permanent or transient, within the given scope.

Multicast scope: The scope field indicates the scope of the IPv6 internetwork for which the multicast traffic is intended. The size of this field is 4 bits. In addition to information provided by multicast routing protocols, routers use multicast scope to determine whether multicast traffic can be forwarded. For multicast addresses there are 14 possible scopes (some are still unassigned), ranging from interface-local to global (including both link-local and site-local).

The following table lists the defined values for the scope field:

Table 12. Multicast scope field values

Value	Scope
0	Reserved
1	Interface-local scope (same node)
2	Link-local scope (same link)
3	Subnet-local scope
4	Admin-local scope
5	Site-local scope (same site)
8	Organization-local scope
E	Global scope
F	Reserved
All other scope field values are currently undefined.	

For example, traffic with the multicast address of FF02::2 has a link-local scope. An IPv6 router never forwards this type of traffic beyond the local link.

Interface-local
The interface-local scope spans a single interface only. A multicast address of interface-local scope is useful only for loopback delivery of multicasts within a node, for example, as a form of interprocess communication within a computer. Unlike the unicast loopback address, interface-local multicast addresses may be joined on any interface.

Link-local
Link-local addresses are used by nodes when communicating with neighboring nodes on the same link. The scope of the link-local address is the local link.

Subnet-local
Subnet-local scope is given a different and larger value than link-local to enable possible support for subnets that span multiple links.

Admin-local
Admin-local scope is the smallest scope that must be administratively configured, that is, not automatically derived from physical connectivity or other, non-multicast-related configuration.

Site-local
The scope of a site-local address is the site or organization internetwork. Addresses must remain within their scope. A router must not forward packets outside of its scope.

Organization-local
This scope is intended to span multiple sites belonging to a single organization.

Global
Global scope is used for uniquely identifying interfaces anywhere in the Internet.

Multicast groups: Group ID identifies the multicast group, either permanent or transient, within the given scope. The size of this field is 112 bits. Permanently assigned groups can use the group ID with any scope value and still refer to the same group. Transient assigned groups can use the group ID in different scopes to refer to different groups. Multicast addresses from FF01:: through FF0F:: are reserved, well-known addresses. Use of these group IDs for any other scope values, with the T flag equal to 0, is not allowed.

All-nodes multicast groups: These groups identify all IPv6 nodes within a given scope. Defined groups include:

- Interface-local all-nodes group (FF01::1)
- Link-local all-nodes group (FF02::1)

All-routers multicast groups: These groups identify all IPv6 routers within a given scope. Defined groups include:

- Interface-local all-routers group (FF01::2)
- Link-local all-routers group (FF02::2)
- Site-local all-routers group (FF05::2)

Solicited-node multicast group: For each unicast address which is assigned to an interface, the associated solicited-node multicast group is joined on that interface. The solicited-node multicast address facilitates the efficient querying of network nodes during address resolution.

Anycast IPv6 Addresses: An IPv6 anycast address is an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the nearest interface), according to the routing protocols' measure of distance. It uses the same formats as a unicast address, so one cannot differentiate between a unicast and an anycast address simply by examining the address. Instead, anycast addresses are defined administratively.

For more information about IPv6 addressing, see *RFC 3513, Internet Protocol Version 6 (IPv6) Addressing Architecture*.

Chapter 2. Transferring Files Using FTP

This chapter describes how to use the File Transfer Protocol (FTP) command and its subcommands to transfer files between your local host and a foreign TCP/IP host. The FTP command and its subcommands, allow you to sequentially access multiple foreign hosts without leaving the FTP environment.

When FTP commands are used with a non-z/VM foreign host, you may need to consult documentation specific to that host and its operating system for information about file naming and supported FTP commands and parameters.

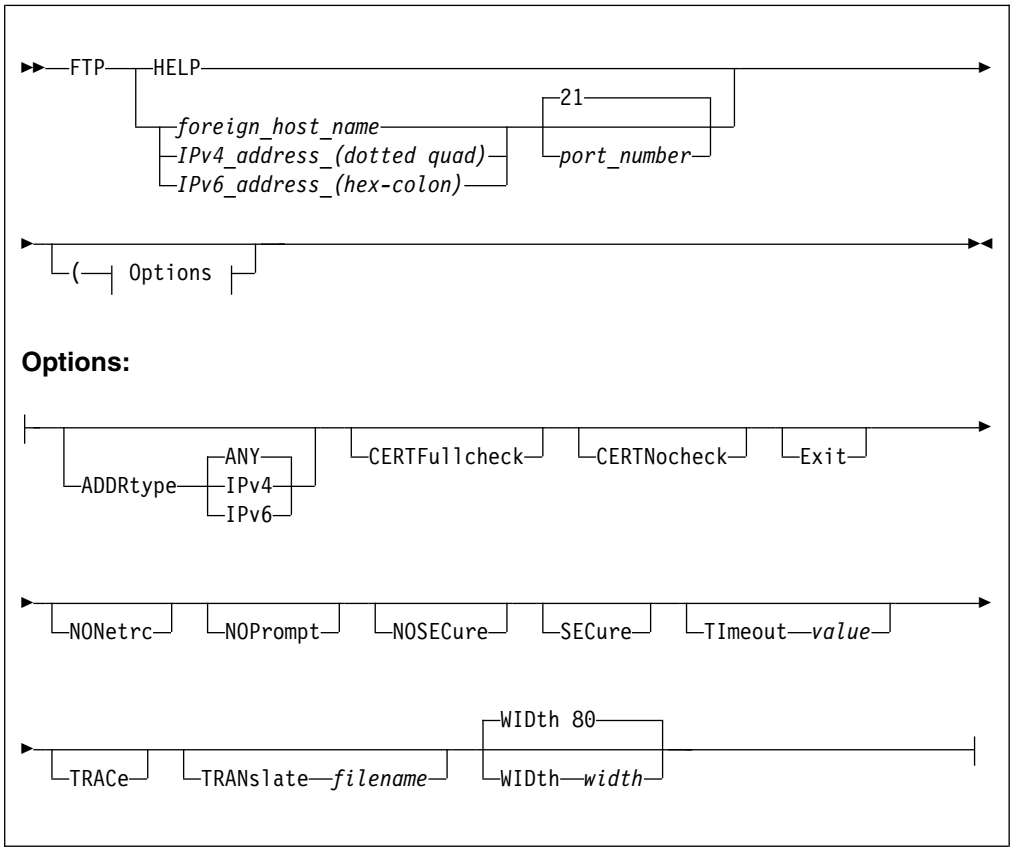
The following table lists general operations and FTP subcommands that correspond to these functions:

Table 13. FTP and Subcommand Functions

FUNCTION	SUBCOMMANDS	
Establish a connection and identify yourself to a foreign host's FTP server	ACCT OPEN USER	NETRC PASS
Obtain status information about FTP on the foreign host	DEBUG NOOP SYSTEM	STATUS
List or work with directories belonging to the foreign host	CD or CWD CDUP LS	DIR MKDIR RMDIR PWD
List or work with directories on the local host	LCD	LPWD
Prepare the environment prior to transferring files	HANGEUL JIS78KJ KSC5601 TCHINESE	EUCKANJI IBMKANJI JIS83KJ LOCSITE SIZE SJISKANJI SUNIQUE TYPE
Transfer Parameter Commands	ASCII EBCDIC BINARY MODE PORT PASSIVE	SENDPORT SENDSITE STRUCT
Perform Special Function	QUOTE	SITE
Work with and transfer files to and from the foreign host	APPEND GET PUT	DELIMIT MGET MPUT
Delete or rename files on the foreign host	DELETE RENAME	MDELETE
Communicate with the underlying operating systems	CMS	
Obtain help about FTP and its subcommands	HELP	

© Copyright IBM Corp. 1987, 2016

FTP Command



Purpose

Use the FTP command to establish an environment, or shell, that allows you to transfer files between your local host and a foreign TCP/IP host.

Once an FTP connection has been established with a foreign host, the various subcommands listed in Table 14 on page 43 can be used to transfer files and interact with the remote FTP server.

Operands

foreign_host_name
: The name of the foreign host to which you are connecting. Specify *foreign_host_name* using an internet host name or an internet address. You are prompted for a host name if this operand is omitted when the FTP command is issued.

IPv4_addres_(dotted quad)
: "Old" style IP address in dotted quad notation. For example, "123.45.67.89".

IPv6_address_(hex-colon)
: "New" style IP address in hex-colon notation. For example, "1234:5678::ABCD:EF09".

port_number
: the port number of the FTP server on the foreign host. The default is port 21, which is considered to be a "well-known" port.

Options

ADDRtype

ANY
Any target host address allowed.

IPv4
Target host address must be in AF_INET (IPv4) address family.

IPv6
Target host address must be in AF_INET6 (IPv6) address family.

CERTFULLCHECK
The remote server certificate and any associated CA certificates are verified for correctness, for control and data connections. If a certificate error is detected, the corresponding connection fails.

A self-signed certificate cannot be used for secure connections unless it has been added to the local certificate database. For more information, see Using Self-Signed Certificates in *z/VM: TCP/IP Planning and Customization*.

CERTNOCHECK
The CERTNOCHECK option is accepted and ignored. All connections are handled as described for the CERTFULLCHECK operand.

Exit
Causes FTP to terminate when an error condition is encountered. Error conditions associated with FTP subcommands will also cause FTP to terminate when this operand is used. For more information about FTP return codes, see "FTP Return Codes" on page 105.

NOSECure
Causes the FTP client not to attempt to secure data and control connections using TLS. The NOSECURE option overrides FTP DATA SECURECONTROL and SECUREDATA statement value of YES. This setting may be overridden while in the FTP session by using the CPROTECT and PRIVATE subcommands.

NONetrc
Suppresses use of the NETRC DATA file. See "The NETRC DATA File" on page 31 for more information.

NOPrompt
Suppresses prompts for the logon user name and password. Use this option when a NETRC DATA file is used and command input is to be obtained through non-interactive means, such as from an exec. See "The NETRC DATA File" on page 31 for more information.

SECure
Causes the FTP client to attempt to secure data and control connections using TLS. The SECURE option overrides FTP DATA SECURECONTROL and SECUREDATA statement value of NO. This setting may be overridden for data connections while in the FTP session by using the CLEAR subcommand.

TImeout *seconds*
Specifies the number of seconds to be used for **all** of the following timing parameters:
- MyopenTime
- DconnTime
- CconnTime
- InactTime

- DataCtTime

The value specified with the TIMEOUT operand is applied to each previously listed timing parameter. If individual timeout values are required, these can be specified in an FTP data file. For more information about this file and timeout parameters and defaults, see "The FTP DATA File" on page 31.

Numeric values between 15 and 720 are accepted for the TIMEOUT parameter.

Note: If the TIMEOUT operand value is not valid, FTP ignores that value and uses the default values established for each timeout parameter.

TRACe
Starts the generation of tracing output. TRACE is used to assist in debugging. Trace data is written to the console.

TRANslate *filename*
The file name of a translation table file other than a standard table. The file type is TCPXLBIN, and the file mode is an asterisk (*). If this parameter is not specified, the FTP command searches sequentially for FTP TCPXLBIN and STANDARD TCPXLBIN. If neither is found, FTP uses the compiled translation table.

This table provides client side translation. For more information on how to configure the translation performed by the z/VM FTP server, see *z/VM: TCP/IP Planning and Customization*. You can also control the translation by using the TRANslate operand of the SITE subcommand. For more information, see "SITE" on page 93.

Note: The FTP TCPXLBIN file is not supplied, because the standard translation table is adequate for most uses. You can create your own FTP TCPXLBIN file if your installation needs a translation for FTP that differs from the translation supported by STANDARD TCPXLBIN. For more information on translation tables see Chapter 15, "Using Translation Tables," on page 437.

If LOADDBCSTABLE is specified in FTP DATA, then *filename* is used to determine which DBCS translation table to load. For information on the loading and customizing of DBCS translation tables, see *z/VM: TCP/IP Planning and Customization*.

WIDth width
Specifies the maximum width to use for lines written to the console by FTP. Data that is longer than the specified *width* is wrapped to successive lines, as necessary. The minimum acceptable width is 1, while the maximum is 32767. The default for *width* is 80. When a FILEDEF is used to control FTP command output, the WIDTH option is ignored.

Usage Notes

- If the specified *foreign_host* is not accessible, or this operand has not been correctly specified, the FTP subcommand environment is established, but no foreign host connection exists. When this occurrs, the OPEN, USER, PASS, and ACCT subcommands can be used to establish a connection and log in as desired.
- If FTP can establish a connection to the specified foreign host **and** a valid entry for that host is present in a NETRC DATA file, by default, FTP uses that host's user name and password to logon to that host. Otherwise, you are prompted for this information, unless the NOPROMPT command option has been specified to suppress prompting.

- Once in the FTP environment, FTP subcommands are limited to 130 characters unless line continuation is used. For more information see "Continuing Subcommand Input Strings" on page 45.

Examples

The example that follows shows information that is typically displayed when an FTP connection is successfully established with a foreign z/VM TCP/IP host. In this example GDLVM7 (in the domain ENDICOTT.IBM.COM) is the *foreign_host* to which the connection is made.

```
Ready;
ftp gdlvm7
VM TCP/IP FTP level 640
Connecting to GDLVM7 9.117.32.29, port 21
220-FTPSERVE IBM VM Level 330 at GDLVM7.ENDICOTT.IBM.COM,
10:04:05 EST THURSDAY 2001-12-01
220 Connection will close if idle for more than 5 minutes.
USER (identify yourself to the host):
terix
>>>USER terix
331 Send password please.
Password:

>>>PASS ********
230-TERIX logged in; working directory = TERIX 191 (ReadOnly)
230 write access currently unavailable
Command:
```

The NETRC DATA File

The NETRC DATA file provides an alternative to responding to FTP prompts for logon information when you connect to a foreign host. When a user name and password for a specific host are defined in this file, FTP will use those values instead of prompting for this information.

When the FTP command or the OPEN subcommand is issued, FTP will search the NETRC DATA file for the first valid match on the specified host. If found, that host's user name and password will be used when a connection to that host is attempted. If no match is found for the host in question, you will be prompted for a user name and password, unless the NOPROMPT option has been specified.

For more information on the NETRC DATA File and how it may be used for other applications, see Appendix B, "Using the NETRC DATA File," on page 453.

The FTP DATA File

The FTP DATA configuration file defines operational characteristics that affect FTP connections and DBCS data translation.

A sample FTP data file is provided as FTP SDATA on the TCPMAINT 592 disk.

Note:

1. If the TIMEOUT operand is specified upon invocation of the FTP command, its corresponding value overrides **all** timing values specified within the FTP DATA file. However, if the TIMEOUT-specified value is not valid, the default timing values identified in the next section are used.

2. When DBCS file transfers are performed, both the remote FTP server and the local client must have the appropriate translate tables loaded.

Statement Syntax

Within FTP DATA, blanks and record boundaries are used to delimit tokens. All characters to the right of (and including) a semicolon are treated as comments.

FTP DATA File Statements

Configuration statements that can be specified in the FTP DATA file are described in this section.

CCONNTIME Statement

The CCONNTIME statement specifies the timeout value used when waiting for a response to a connection close request on a Control connection.

Operands

seconds

The number of seconds to allow before a timeout when waiting for a response to a connection close request on a Control connection. The minimum CCONNTIME value is 15 and the maximum is 720; the default is 30 seconds.

DATACTTIME Statement

The DATACTTIME statement specifies the timeout value used when waiting for a response to a TCP send or TCP receive request.

Operands

seconds
The number of seconds to allow before a timeout when waiting for a response to a TCP send or TCP receive request. The minimum DATACTTIME value is 15 and the maximum is 720; the default is 120 seconds.

DCONNTIME Statement

The DCONNTIME statement specifies the timeout value used when waiting for a response to a connection close request on a Data connection.

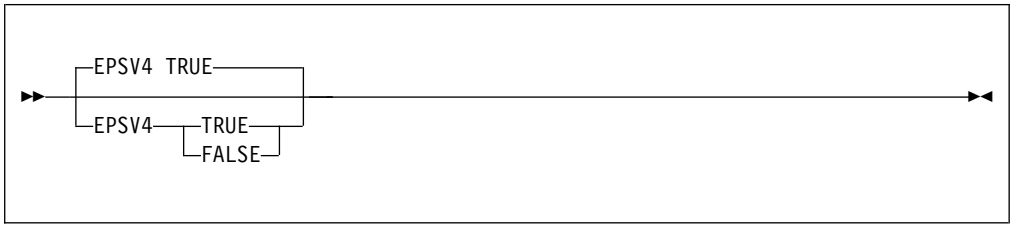
Operands

seconds

The number of seconds to allow before a timeout when waiting for a response to a connection close request on a Data connection. The minimum DCONNTIME value is 15 and the maximum is 720; the default is 120 seconds.

EPSV4 Statement

The EPSV4 statement specifies if FTP client will use EPSV or EPRT command to establish the connection with the server for data transfer.



Operands

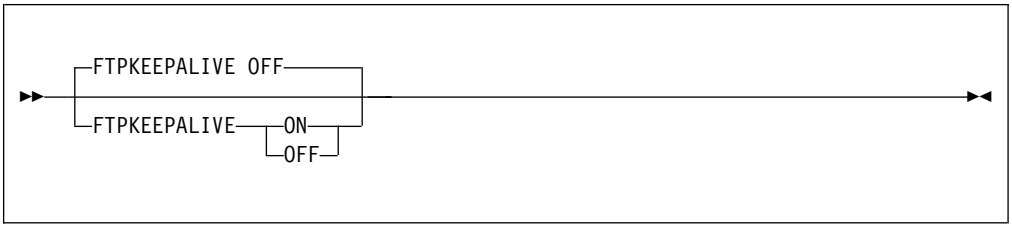
TRUE
Specifies that the FTP client will first try to use a EPSV or EPRT command to establish the connection with the server for data transfer. If the server rejects the EPSV or EPRT command, the client will then try to establish the data connection using the PASV or PORT command.

Note: If the server rejects either the EPSV or the EPRT command during the session, the client won't send EPSV or EPRT to the server again, even when EPSV4 is specified.

FALSE
Specifies that the FTP client will send the server a PASV or PORT command to establish the connection with the server for data transfer.

FTPKEEPALIVE Statement

The FTPKEEPALIVE statement specifies if FTP client will use TCP/IP stack's keepalive timer value for control connection.



Operands

ON Specifies that the FTP client should make use of the TCP/IP server's keepalive mechanism to avoid timing out an idle control connection. The frequency with which packets are sent is determined by the KEEPALIVEOPTIONS statement in the TCP/IP server's configuration file.

OFF

Specifies that the FTP client should allow idle control connections to time out. OFF is the default.

INACTTIME Statement

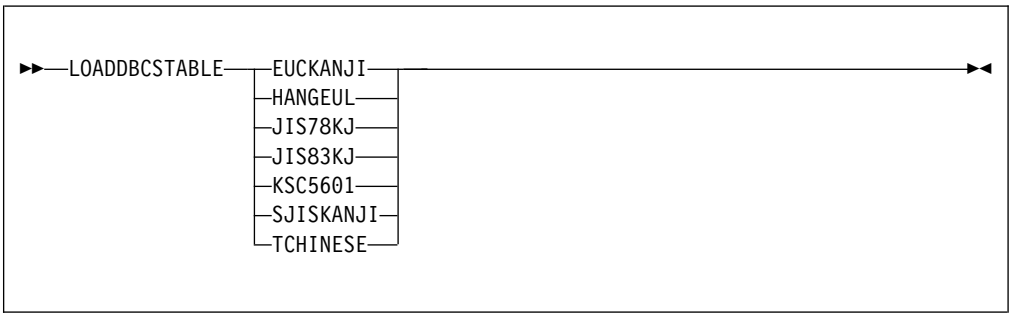
The INACTTIME statement specifies the timeout value used when attempting to establish a data connection to complete FTP subcommand operations.

Operands

seconds
The number of seconds to allow before a timeout when attempting to establish a data connection to complete FTP subcommand operations. The minimum INACTTIME value is 15 and the maximum is 720; the default is 120 seconds.

LOADDBCSTABLE Statement

The LOADDBCSTABLE statement indicates to the FTP client which DBCS translate tables should be loaded at initialization time. By using multiple LOADDBCSTABLE parms, any number of translate tables may be selected ranging from none to all of the DBCS translate tables.



Operands

JIS78KJ
Indicates that the JIS 1978 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table file.

JIS83KJ
Indicates that the JIS 1983 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table file.

SJISKANJI
Indicates that the Shift JIS Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table file.

EUCKANJI
Indicates that the Extended Unix Code Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table file.

HANGEUL
Indicates that the Hangeul DBCS translation table should be loaded from the TCPHGBIN binary translate table file.

KSC5601
Indicates that the Korean Standard Code KSC-5601 DBCS translation table should be loaded from the TCPHGBIN binary translate table file.

TCHINESE
Indicates that the Traditional Chinese (5550) DBCS translation table should be loaded from the TCPCHBIN binary translate table file.

If the LOADDBCSTABLE parameter is not specified, specified incorrectly, or if FTP DATA is not accessible, then no DBCS translate tables will be loaded, and the corresponding FTP client DBCS transfer types will be unavailable.

Additional virtual storage may be required by the FTP client if a large number of translate tables are loaded concurrently.

Note: The IBMKANJI transfer type does not require any translate table to be loaded. For more information on loading and customizing DBCS translate tables, see *z/VM: TCP/IP Planning and Customization*.

MYOPENTIME Statement

The MYOPENTIME statement specifies the timeout value used when attempting to establish a connection.

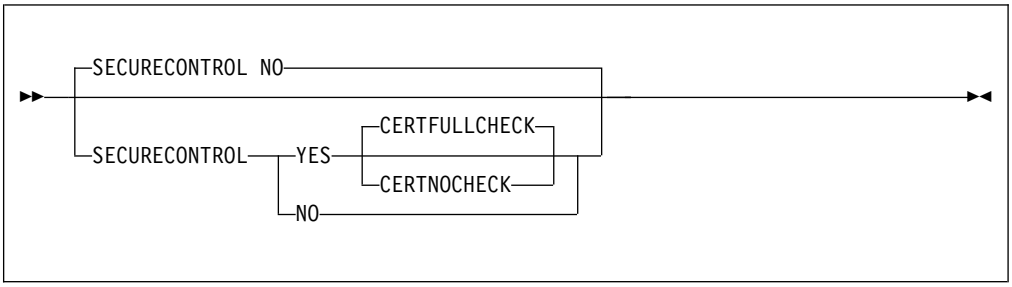
Operands

seconds
The number of seconds to allow before a timeout when attempting to establish a connection. The minimum MYOPENTIME value is 15 and the maximum is 720; the default is 60 seconds.

SECURECONTROL Statement

The SECURECONTROL statement specifies if the FTP client will secure control connections using TLS.

Note: In order to secure FTP data connections, the control connection associated with the FTP session must first be secured. See "Transferring Files Using Secure FTP" on page 57 for more information.



Operands

NO Specifies the FTP client will not secure control connections using TLS. This is the default. When NO is specified, the FTP user may optionally override the SECURECONTROL setting at initialization by using the SECURE option to the FTP command or while in the FTP session by using the CPROTECT subcommand.

YES
Specifies the FTP client will secure control connections using TLS. When YES is specified, the FTP user may optionally override the SECURECONTROL setting at initialization by using the NOSECURE option to the FTP command.

CERTFULLCHECK
When specified with YES, the remote server certificate and any associated CA certificates are verified for correctness for control connections. If a certificate error is detected, the corresponding connection fails.

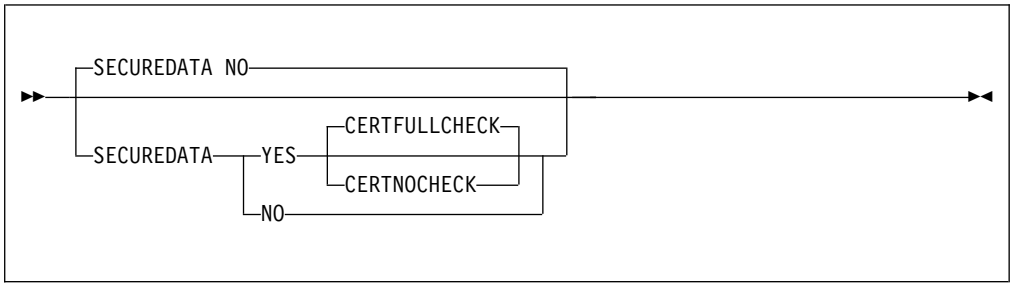
A self-signed certificate cannot be used for secure connections unless it has been added to the local certificate database. For more information, see Using Self-Signed Certificates in *z/VM: TCP/IP Planning and Customization*.

CERTNOCHECK
The CERTNOCHECK option is accepted and ignored. All connections are handled as described for the CERTFULLCHECK operand.

SECUREDATA Statement

The SECUREDATA statement specifies if the FTP client will secure data connections using TLS.

Note: In order to secure FTP data connections, the control connection associated with the FTP session must first be secured. See "Transferring Files Using Secure FTP" on page 57 for more information.



Operands

NO Specifies the FTP client will not secure data connections using TLS. This is the default. When NO is specified, the FTP user may optionally override the SECUREDATA setting at initialization by using the SECURE option to the FTP command or while in the FTP session by using the PRIVATE subcommand.

YES
Specifies the FTP client will secure data connections using TLS. This is the default. When YES is specified, the FTP user may optionally override the SECUREDATA setting at initialization by using the NOSECURE option to the FTP command or while in the FTP session by using the CLEAR subcommand.

CERTFULLCHECK
When specified with YES, the remote server certificate and any associated CA certificates are verified for correctness for data connections. If a certificate error is detected, the corresponding connection fails.

A self-signed certificate cannot be used for secure connections unless it has been added to the local certificate database. For more information, see Using Self-Signed Certificates in *z/VM: TCP/IP Planning and Customization*.

CERTNOCHECK
The CERTNOCHECK option is accepted and ignored. All connections are handled as described for the CERTFULLCHECK operand.

FTP Subcommands

The FTP subcommands listed in Table 14 can be used to perform file transfer operations between your local z/VM host and a foreign TCP/IP host, once an FTP connection (and thus, the FTP subcommand environment) has been established. For information about establishing an FTP connection, see "FTP Command" on page 28.

Table 14 provides a summary of all FTP subcommands that can be used with the z/VM FTP server and client, and shows the shortest abbreviation, a brief description, and a page reference for more information for each subcommand that is supported.

Table 14. FTP Subcommands

Subcommand	Minimum Abbreviation	Description	Location
ACCT	AC	Sends host-dependent account information.	"ACCT" on page 58
APPEND	AP	Appends a file on your local host to a file on the foreign host.	"APPEND" on page 59
ASCII	AS	Sets the transfer type to ASCII.	"ASCII" on page 59
BINARY	B	Sets the transfer type to IMAGE.	"BINARY" on page 60
CCC	CCC	Causes the FTP client to clear (stop using TLS on) the control connection.	"CCC" on page 61
CD	CD	Changes the working directory. CWD is a synonym for CD.	"CD or CWD" on page 62
CDUP	CDU	Changes the working directory to the parent directory on the foreign host. CD is a synonym for CDUP.	"CDUP" on page 65
CLEAR	CLE	Causes the FTP client to set up for clear data connections when set up for secure data connections using TLS.	"CD or CWD" on page 62
CLOSE	CLO	Disconnects from the foreign host.	"CLOSE" on page 66
CMS	CM	Passes a CMS command.	"CMS" on page 66
CPROTECT	CP	Causes the FTP client to attempt to secure the control connection using TLS.	"CD or CWD" on page 62
CWD	CW	Changes the working directory. CD is a synonym for CWD.	"CD or CWD" on page 62
DEBUG	DEB	Toggles internal debug options.	"DEBUG" on page 67
DELETE	DELE	Deletes a single file on the foreign host.	"DELETE" on page 68
DELIMIT	DELI	Displays the delimiter character between the file name and the file type.	"DELIMIT" on page 68
DIR	DI	Lists the directory entries for files on the foreign host. LIST is a synonym for DIR.	"DIR" on page 69
EBCDIC	EB	Sets the transfer type to EBCDIC.	"EBCDIC" on page 70
EUCKANJI	EU	Sets the transfer type to EUCKANJI.	"EUCKANJI" on page 71
GET	G	Copies a file from the foreign host to your local host. RETRIEVE is a synonym for GET.	"GET" on page 71
HANGEUL	HA	Sets the transfer type to HANGEUL.	"HANGEUL" on page 72

Table 14. FTP Subcommands (continued)

Subcommand	Minimum Abbreviation	Description	Location
HELP	H	Displays help information for FTP.	"HELP" on page 73
?	?	Provides information to use FTP.	"HELP" on page 73
IBMKANJI	I	Sets the transfer type to IBMKANJI.	"IBMKANJI" on page 73
JIS78KJ	JIS7	Sets the transfer type to JIS78KJ.	"JIS78KJ" on page 74
JIS83KJ	JIS8	Sets the transfer type to JIS83KJ.	"JIS83KJ" on page 74
KSC5601	K	Sets the transfer type to KSC5601.	"KSC5601" on page 75
LCD	LC	Changes the working directory on the local host.	"LCD" on page 75
LOCSITE	LOCSI	Changes local site information.	"LOCSITE" on page 76
LOCSTAT	LOCST	Displays FTP status information for the local host.	"LOCSTAT" on page 77
LPWD	LP	Displays the name of the active working directory on the local host.	"LPWD" on page 78
LS	LS	Lists the names of files on the foreign host. NLST is a synonym for LS.	"LS" on page 78
MDELETE	MD	Deletes multiple files on the foreign host.	"MDELETE" on page 80
MGET	MG	Copies multiple files from the foreign host to your local host.	"MGET" on page 81
MKDIR	MK	Creates a new directory on the foreign host.	"MKDIR" on page 81
MODE	MO	Specifies the mode or data format of the transfer.	"MODE" on page 82
MPUT	MP	Copies multiple files on your local host to the foreign host.	"MPUT" on page 83
NETRC	NE	Enables or disables automatic logon capability through use of a NETRC DATA file.	"NETRC" on page 84
NOOP	NO	Checks whether the foreign host is still responding.	"NOOP" on page 84
OPEN	O	Opens a connection to a foreign host. CONNECT is a synonym for OPEN.	"OPEN" on page 85
PASS	PA	Supplies a password to the foreign host.	"PASS" on page 86
PASSIVE	PASSI	Controls whether the client or server establishes connections for data transfers.	"PASSIVE" on page 86
PRIVATE	PR	Causes the FTP client to set up for secure data connections using TLS.	"CD or CWD" on page 62
PUT	PU	Copies a file on your local host to the foreign host.	"PUT" on page 88
PWD	PW	Displays the name of the active working directory on the foreign host.	"PWD" on page 89
QUIT	QUI	Leaves the FTP command environment.	"QUIT" on page 90
QUOTE	QUO	Sends an uninterpreted string of data.	"QUOTE" on page 90

Table 14. FTP Subcommands (continued)

Subcommand	Minimum Abbreviation	Description	Location
RENAME	REN	Renames a file on the foreign host.	"RENAME" on page 91
RMDIR	RM	Remove a directory from the foreign host.	"RMDIR" on page 91
SENDPORT	SENDP	Enables or disables automatic transmission of the FTP server PORT subcommand. TOGLPORT is a synonym for SENDPORT.	"SENDPORT" on page 92
SENDSITE	SENDS	Enables or disables automatic transmission of the SITE subcommand.	"SENDSITE" on page 93
SITE	SI	Sends information to the foreign host using site-specific commands.	"SITE" on page 93
SIZE	SIZE	Retrieves the transfer size (in bytes) for a foreign host file.	"SIZE" on page 96
SJISKANJI	SJ	Sets the transfer type to SJISKANJI.	"SJISKANJI" on page 96
STATUS	STA	Displays status information for the foreign host.	"STATUS" on page 97
STRUCT	STR	Sets the file transfer structure.	"STRUCT" on page 97
SUNIQUE	SU	Toggles the storage methods.	"SUNIQUE" on page 98
SYSTEM	SY	Displays the name of the foreign host's operating system.	"SYSTEM" on page 99
TCHINESE	TC	Sets the transfer type to TCHINESE.	"TCHINESE" on page 100
TYPE	TY	Specifies the transfer type.	"TYPE" on page 100
USER	U	Identifies you to a foreign host. LOGIN is a synonym for USER.	"USER" on page 101

Continuing Subcommand Input Strings

Due to system limitations, FTP subcommand input lines are restricted to 130 characters; subcommand text that is present after 130 characters is ignored. When FTP subcommands are supplied, a continuation operator — a plus sign (+) preceded by a single blank — can be used to continue a string on a subsequent input line. The plus sign and its preceding blank will not be present in the resulting final string; thus, any blanks necessary to delimit multiple tokens for certain FTP subcommands must be explicitly provided as part of the input string(s) that are continued. Delimiting blanks can be included either before the " +" continuation operator, or at the beginning of the next line of continued text. Note that multiple blanks are not significant (that is, are not preserved). The limit for the subcommand input stream continued in this fashion is 200 bytes.

Example of Line Continuation

Suppose the current working directory on the foreign host is:

```
GPLSRV2:DOC.LIBRARY.PROJECTX
```

The following MKDIR subcommand is then issued, using two lines:

```
mkdir gplsrv2:doc.library.projectx. +
worlfiles
```

The foreign host replies with:

```
>>>MKD gplsrv2:doc.library.projectx.worlfiles
257 New directory GPLSRV2:DOC.LIBRARY.PROJECTX.WORLFILES has
    been created.
```

After creating several other directories, you realize the `worlfiles` directory should be `workfiles`. To correct this, you issue an RMDIR command of the form:

```
rmdir  +
```

with the remainder of the command completed by retrieving and using previously entered commands:

```
gplsrv2:doc.library.projectx. +
worlfiles
```

The foreign host replies with:

```
>>>RMD gplsrv2:doc.library.projectx.worlfiles
250 Directory GPLSRV2:DOC.LIBRARY.PROJECTX.WORLFILES has been
    removed.
```

You then correct the directory name, again retrieving and using previously entered commands (with the last one corrected to `workfiles`):

```
mkdir  +
gplsrv2:doc.library.projectx. +
workfiles
```

The response from the foreign host is then:

```
>>>MKD gplsrv2:doc.library.projectx.workfiles
257 New directory GPLSRV2:DOC.LIBRARY.PROJECTX.WORKFILES has
    been created.
```

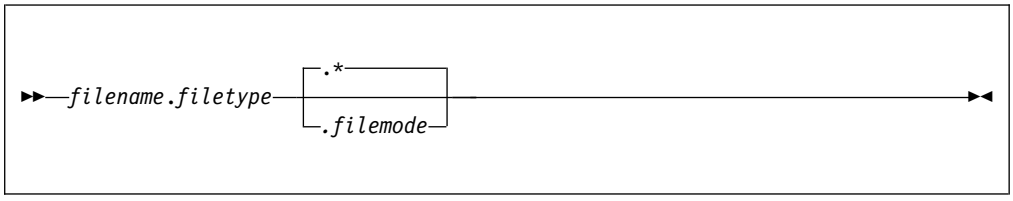
File Name Formats

FTP provides a mechanism to transfer files from one host to another, in which different operating systems (and thus, file systems) may be in use. Therefore, the format used to identify a file is dependent on the host system where that file resides or will reside.

In FTP subcommands used to manipulate files, it is necessary to distinguish files associated with the local host from those on a foreign host. In the FTP subcommand descriptions throughout this chapter, files that are specific to the local host are identified using the term *localfile*, whereas a file that is specific to a remote (foreign) host is identified using the term *foreignfile*.

Working with local CMS files

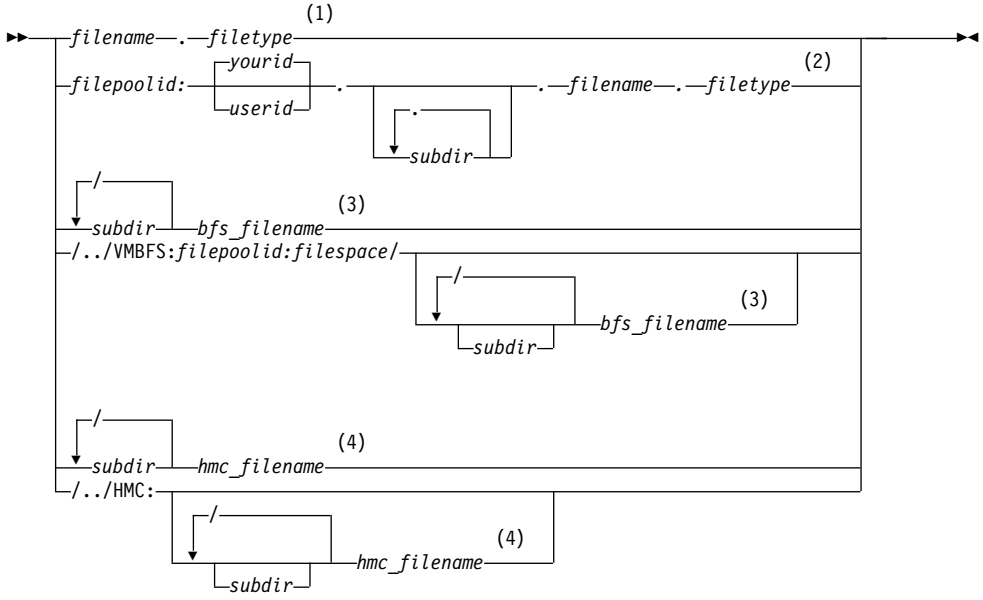
When CMS files are identified for FTP subcommands that require a local file (*localfile*), use this naming format:



Note:

1. When a specific file mode is provided for a CMS local file, only the resource accessed at that file mode is used to obtain or create the file that has been identified.
2. When a file mode is not specified, or is specified as an asterisk (*) for a CMS local file, the resources involved in manipulation of the referenced file may vary, based on the FTP subcommand in use. For certain FTP subcommands, a file mode specified as an asterisk signifies the local working directory *only*, whereas for others, this may imply use of the local working directory *first*, followed by the current CMS search order.

Working with Remote Files: When z/VM is in use on the foreign host and a *foreignfile* name is required, use the naming format that follows:



Notes:

1 Format for minidisk and reader files.

2 Format for SFS files.

3 Formats for BFS files.

4 Formats for HMC files.

To reference minidisk or reader files, the appropriate working directory must be established on the foreign VM host. Fully-qualified SFS, BFS, and HMC foreign files can be referenced regardless of whether they reside in the current working directory on the foreign host.

For more information about CMS file names and limitations, see the *z/VM: CMS Commands and Utilities Reference*. For details regarding BFS file-naming conventions, refer to the *z/VM: OpenExtensions User's Guide*.

File Name Pattern Matching

When subcommands are issued that affect multiple CMS files, such as MDELETE, MGET, and MPUT, pattern matching can be used to affect the scope of the set of

files affected by these subcommands (as can be done with the CMS LISTFILE command). Pattern matching is accomplished by specifying an asterisk (*) as a portion of, or in place of, *filename* or *filetype* when a file name is specified for these commands.

Pattern matching can also be used with the *name* operand that is used in DIR and LS subcommands. In addition to the pattern matching just described for *filename* and *filetype* (when *name* is specified using the format *filename.filetype*), the *name* operand itself can be used for pattern matching. Pattern matching based on the *name* operand is accomplished by specifying an asterisk (*) as a portion of, or in place of, the *name* operand.

Note:

1. When the foreign host working directory is a virtual reader, pattern matching characters that are used as a portion of the *filename, filetype* or *name* operands must be specified as a leading or trailing character. An asterisk (*) can still be used in place of these operands.

File Name Case Considerations

When FTP operations are performed between hosts that are based on differing file system implementations, it may be necessary to account for differences that can arise due to **case sensitive** file naming conventions.

For example, when multiple CMS files are identified for MDELETE, MGET, and MPUT operations, the name and type of these files are provided by the z/VM FTP server to a connected client in **lowercase**, whereas for other subcommands (such as DIR or LS) these same files may be identified using uppercase names. For subcommands that affect only a single file (such as PUT or GET), case is preserved for file naming information that is transmitted.

While these response differences do not adversely affect operations between z/VM hosts, they may affect file-related actions performed on a connected host that is case-sensitive (with respect to naming conventions and the underlying file system implementation of that host).

Note: For minidisks and SFS directories, CMS files are created with names in **uppercase**. FTP on z/VM does not support mixed-case file naming for these resources.

Working with Non-CMS Files

For detailed information about naming and referencing files on a non-z/VM host, consult the documentation specific to that host and its operating system. The information provided in Appendix A, “Specifying Files and Data Sets,” on page 449 may be useful as well.

In general, the z/VM FTP client does not restrict the naming of a *foreignfile* when such information is supplied to the FTP server on a foreign host. For example, the slash (/) often used in naming Unix files can be used when these files are referenced, even though this is not a valid CMS file name character. However, length restrictions may at times be encountered, due to implementation considerations.

Information about how certain naming conditions and problems are dealt with by FTP on a z/VM host follows.

Many commonly-referenced file systems (Unix is one example) maintain files using a descriptive file *name* only, and have no underlying support for a file *type*, as does CMS. When these files are transferred to a z/VM host, and the existing foreign file name cannot be used to adequately produce a CMS file name and file type, that file is stored on CMS minidisks and SFS directories using a CMS file type of **$DEFAULT**.

For some files, due to their naming, a specific CMS file name or file type must be provided when those files are retrieved from a foreign host. For example, to retrieve a *foreignfile* that begins with a period (.), the *foreignfile* specified with the GET subcommand must include an explicit *filename*. To retrieve a group of such files using an MGET subcommand, use an asterisk (*) as the *foreignfile* file name.

When the remote host is the z/VM FTP server, file identifiers (including directory and file name) for files residing in the HMC removable media are limited to 191 characters.

Quoted Strings and Embedded Spaces (Blanks)

When the need arises to specify an FTP subcommand that includes a foreign file or directory name that requires spacing, that name should be enclosed using single (') quotation marks.

When the z/VM FTP client encounters a subcommand operand string that is enclosed within quotation marks *and* that string contains embedded spaces, it discards the delimiting quotation marks *before* the string is passed to the FTP server on the foreign host.

If quotation marks are encountered that enclose a subcommand string which does *not* contain embedded blanks, the z/VM FTP client assumes the supplied quotation marks are inherent to this string. Thus, the delimiting quotation marks are retained, and are passed on to the FTP server on the foreign host.

Fully-Qualified Pathnames

For certain FTP subcommands, the z/VM FTP server allows a fully-qualified pathname.

For example, assume the SFS directory FPSERV1:TSTUSER.TESTCASES is the working directory, but that information about all **TEST007** files that reside in the FPSERV1:TSTUSER.TESTDATA directory needs to be obtained at a given time. This information can be obtained using the DIR command that follows:

```
dir fpserv1:tstuser.testdata.test007.*
```

If these files reside on the NOTSECUR 191 minidisk (for which a read password of ALL is in effect), this information would be retrieved using this command:

```
dir notsecur.191/test007.*
```

A fully-qualified pathname can be specified for these FTP subcommands:

- APPEND (APPE)
- DELETE (DELE)
- DIR (LIST)
- LS (NLST)
- GET, MGET (RETR)
- SIZE (SIZE)
- PUT, MPUT (STOR)
- PUT, MPUT with SUNIQUE active (STOU)

- RENAME

Note: When fully-qualified path names are used with FTP subcommands, the FTP server temporarily acquires the appropriate z/VM file system resource necessary to satisfy a request. This action by the FTP server does not alter the working directory that has been established through use of either the CD or CWD commands.

Interpretation of Path Information

The z/VM FTP server accepts path information for FTP commands such as CD and GET. The path information can be specified in one of three formats:

- full qualified path
- absolute path
- relative path

Because the z/VM FTP server provides access to different file systems (such as SFS and BFS), a fully qualified path is different than an absolute path.

The **full qualified path** specifies the file system and the path. It can address any given file or directory accessible on the system.

For example:

```
/../VMBFS:SERVT2:TCPBFS4/SUBDIR1/SUBDIR2
```

The **absolute path** contains an absolute path on the current file system, but not the file system itself. It makes use of the file system of the current working directory (as established by either a CD or CWD subcommand). Hence it can only address files and directories on the current file system.

For example:

```
/SUBDIR1/SUBDIR2
```

The **relative path** specifies a path starting with the current working directory on the current file system. It makes use of the file system of the current working directory (as established by either a CD or CWD subcommand). Hence, it can only address files and directories on the current file system.

For example:

```
SUBDIR1/SUBDIR2
```

As a practical example consider the following BFS directory structure:

```
SERVT2:TCPBFS4
|- FILE1.BIN
|- SUBDIR1
   |
   |- FILE2.BIN
   |- SUBDIR2
      |- FILE3.BIN
```

Assume you want to retrieve FILE3.BIN:

- Using a **full qualified path** the file can be retrieved independent from what the current directory is with the command:

  ```
  GET /../VMBFS:SERVT2:TCPBFS4/SUBDIR1/SUBDIR2/FILE3.BIN
  ```

  This command will work no matter what the current file system or path is.

- Using an **absolute path** you first need to change the working directory to the SERVT2:TCPBFS4 BFS file pool. Note that we directly change into the SUBDIR1 directory:

  ```
  CD /../VMBFS:SERVT2:TCPBFS4/SUBDIR1
  ```

  Now you can retrieve the file using an absolute path with the following command:

  ```
  GET /SUBDIR1/SUBDIR2/FILE3.BIN
  ```

  This command will work with any current directory on the SERVT2:TCPBFS4 file pool.
- From inside the working directory established using the CD command above you can also specify a **relative path** to retrieve FILE3.BIN:

  ```
  GET SUBDIR2/FILE3.BIN
  ```

  This command does not only require the current working directory to be on the correct file system but is also dependent of the current position within the file system.

Storage Space Requirements for Transferred Files

The information that follows must be considered when files are stored on a z/VM host, especially when minidisk or SFS storage space is constrained, or when attempts are made to minimize the use of such space.

As part of necessary overhead required by the FTP server and by CMS for performing file system management, a number of storage blocks must be reserved and used for other than data storage purposes. For instance, some blocks are used as control blocks to maintain the minidisk file directory (for which additional block space may be required as new files are created), while others are used as pointer blocks to manage file structure.

However, these various *reserved* blocks are not reflected in the output returned for the CMS QUERY DISK or QUERY LIMITS commands. This means the results obtained from these commands must be used only as an *approximation* of storage block usage and availability when determining whether a file can be stored on a z/VM host.

File Transfer Methods

When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved. The nature of a file transfer is primarily controlled using two FTP subcommands:

- the MODE subcommand, which determines how the file contents are transmitted. For more information, see "MODE" on page 82.
- the TYPE subcommand, which establishes attributes of the file contents. For more information, see "TYPE" on page 100.

Controlling File Translation

Because z/VM maintains data in EBCDIC format, it is often necessary to be aware of and have control over how EBCDIC-ASCII file translation is performed when files are transferred to or from a remote host.

Table 15 on page 52 shows recommendations for setting transmission attributes for file transfers that are performed between differing host systems. In this table, IBM host systems (VSE/ESA, z/VM or OS/390® hosts) are referred to as "EBCDIC" hosts. By comparison, a Unix, Windows, or Linux host is considered to be an

"ASCII" host. With respect to EBCDIC hosts, text data is considered to be comprised of only standard, displayable characters, whereas for an ASCII host, text data generally contains the carriage return (ASCII X'0D' and EBCDIC X'15'), the line feed (ASCII X'0A' and EBCDIC X'25') and displayable characters.

Table 15. Recommended Methods of File Transfer

Source Host	Intermediate Host	Target Host	Data	TYPE Setting	Mode Setting
EBCDIC	(none)	EBCDIC	binary	**E** (EBCDIC)	**B** (Block)
EBCDIC	(none)	EBCDIC	text	**E** (EBCDIC)	**S** (Stream)
EBCDIC	(none)	ASCII	text	**A** (ASCII)	**S** (Stream)
ASCII	(none)	EBCDIC	text	**A** (ASCII)	**S** (Stream)
ASCII	(none)	EBCDIC	binary	**I** (Image) (1*)	**S** (Stream)
ASCII	EBCDIC (2*)	ASCII	binary	**I** (Image) (1*)	**S** (Stream)
EBCDIC	ASCII (2*)	EBCDIC (3*)	binary	**I** (Image) (1*)	**S** (Stream)

1. The BINARY subcommand can be used to set the transfer type to Image.
2. Used only for storage purposes.
3. See accompanying notes for more information.

Note:

1. When files are transferred between IBM EBCDIC host systems, the combined use of the MODE B and TYPE E subcommands is generally sufficient to achieve satisfactory results.
2. When a CMS file such as a MODULE, (which has an internal format which must be preserved) is transferred using an intermediate ASCII host, that file should first be compressed using the PACK option of the CMS COPYFILE command. This "packed" file should then be handled as a *binary* file until it is transferred to its final destination. For the z/VM destination host, a SITE (or LOCSITE) FIXRECFM 1024 subcommand must be issued prior to the PUT (or GET) subcommand used to transfer the file — this is necessary to create a file that can then be uncompressed using the UNPACK option of the CMS COPYFILE command.

Automatic File Translation

The z/VM FTP server can be configured by the system administrator to automatically perform EBCDIC-ASCII file translation based on the value of a *file extension*, which is defined to be the last component of a file name — that is, the characters that follow the last period in a path name. For file extensions, up to eight characters are matched and mixed case is **not** respected.

Automatic file translation, when enabled, is applicable to *only* the **Image** (TYPE I) file transfer type. Thus, if the transfer type is changed to other than Image — for example, ASCII — during a session for which automatic file translation is active, automatic translation ceases and files are transferred in accordance with the newly established transfer type. In such a case, automatic file translation can be reinstated by changing back to the **Image** transfer type.

Automatic file translation is recommended when a web browser or graphical FTP client is used to interact with a z/VM FTP server, because these clients often default to using a transfer type of **Image** (or, **binary**) and do not offer a way for a different file transfer type to be specified.

The associations between file extensions and file translation are configured by the z/VM system administrator, as is the initial automatic file translation setting for an FTP session. While file extension associations cannot be changed by an FTP client, whether automatic file translation is performed by the server can be controlled by using the AUTOTRANS operand of the SITE subcommand. For more information, see "SITE" on page 93.

File List Formats

After an FTP connection has been established with a z/VM foreign host, the FTP server can be instructed to provide **DIR** subcommand responses (file *lists*) in one of two different *list formats*:

- VM-format (**VM**), or
- Unix-format (**UNIX**), sometimes referred to as Unix *long-list* format.

These formats are described in more detail in the following sections.

VM-format Lists

The VM-format list response provides file information in a format that — for the minidisk and Shared File System (**SFS**) directory file groups — closely resembles that produced by the CMS LISTFILE command. A sample VM-format list response follows:

```
Command:
dir
>>>PORT 9,117,32,30,4,53
200 Port request OK.
>>>LIST
125 List started OK
ENDTRACE TCPIP    F      80       1      1 1999-07-28 12:24:01 TCM191
LASTING  GLOBALV  V      43      10      1 1999-11-16  9:05:22 TCM191
NOTRACE  TCPIP    F      80       1      1 1999-12-16 13:39:21 TCM191
OBEY     EXEC     V      72     153      2 1996-01-03 16:07:07 TCM191
PACKMODL TESTFILE F    1024     468    117 1996-07-28 13:56:36 TCM191
PROFILE  EXEC     V      30      10      1 1999-11-16  9:01:10 TCM191
RECF80   TESTFILE F      80       5      1 2000-01-17 14:47:19 TCM191
SETX     XEDIT    V      46      13      1 1999-11-04  9:13:53 TCM191
TCPIP    DATA     V      72      99      2 1999-11-18 17:24:05 TCM191
TCPMNT2  NETLOG   V     108      48      2 2000-01-17 14:49:09 TCM191
TCPMNT2  SYNONYM  F      80      10      1 1999-11-10  8:46:12 TCM191
TCPSLVL  EXEC     V      37      23      1 1999-02-05 13:20:46 TCM191
TEST     EXEC     V      71     107      2 1997-07-14 10:43:17 TCM191
TRACE2   TCPIP    F      80       1      1 1999-12-30 11:15:22 TCM191
250 List completed successfully.
Command:
```

For files and directories that are maintained using the z/VM Byte File System (**BFS**), VM-format lists are identical to z/VM OPENVM LISTFILE responses. A sample response for BFS directory information follows:

```
Command:
dir
>>>PORT 9,117,32,30,4,53
200 Port request OK.
>>>LIST
125 List started OK
05/20/2000 13:38:19   F        1          65758 'bfsline.cpy'
05/19/2000 11:02:15   F        1          65758 'bfsline.txt'
06/03/2000 12:27:48   F        1          15414 'bfstest.cpy'
05/20/2000 13:38:05   F        1          15414 'bfstest.output'
05/20/2000 13:38:42   F        1         772902 'bfswork.output'
03/31/2000 15:49:27   F        1         782444 'bfswork.txt'
05/20/2000 13:39:20   F        1          13930 'lotsonl.putdata'
05/19/2000 09:41:21   F        1          13930 'lotsonl.txt'
06/15/2000 09:29:25   F        1            278 'mail.maw'
```

```
05/20/2000 13:39:34   F        1           278 'mail.putdata'
05/20/2000 15:30:45   F        1         13930 'nls.new'
05/20/2000 14:02:24   F        1         13931 'nls.txt'
08/21/2000 10:03:17   F        1           328 'rock.rules'
05/20/2000 13:40:05   F        1            58 'testfil2.putdata'
04/26/2000 14:34:42   F        1            63 'testfil2.txt'
08/21/2000 05:28:40   D        -             - 'ALTERNATE'
12/28/2000 17:36:19   D        -             - 'FIRST'
250 List completed successfully.
Command:
```

For a z/VM virtual reader (**RDR**), list responses are similar to that produced by the CP QUERY RDR ALL command, but with certain fields removed. A sample VM-format response for a virtual reader follows:

```
Command:
dir
>>>PORT 9,117,32,29,10,213
200 Port request OK.
>>>LIST
125 List started OK
0013 SMTP     00000011 1999-11-03 12:46:10 CIBULAPR  MAIL
0154 RSCS     00002789 1999-12-29 10:12:46 FL3XSAMP  TXT
0116 TCPLVL2  00000170 1999-12-16 11:39:07 TCP-HELP  LIST3820
0115 TCPLVL2  00002874 1999-12-16 11:39:06 TCP-OVER  LIST3820
0153 RSCS     00002825 1999-06-29 10:12:45 FL32SAMP  TXT
0214 SMTP     00000015 2000-01-14 12:50:10 CIBULAMA  MAIL
250 List completed successfully.
Command:
```

The fields in this type of response are referred to (in order) as the **spoolid**, **origin**, **records**, **date**, **time**, **filename** and **filetype** fields.

For files residing in the removable media of an HMC, the LISTFORMAT setting for an FTP session is ignored because there is no VM-format list for HMC directories. For HMC directories, Unix-format lists are always sent in response to the DIR subcommand.

Unix-format Lists

When a web browser, or other graphical FTP client is used to interact with a z/VM FTP server, the use of Unix-format lists is recommended. This is because these types of clients have, in general, been implemented to work with a Unix file system model and expect Unix-like information to be presented as FTP transactions are performed. If VM-format file lists are supplied when such a client is used, directory and file information may not be correctly displayed or managed by the client; this may then cause FTP processing capabilities to become limited or adversely affected.

The list format initially supplied by an FTP server is configured by the z/VM system administrator. However, the list format used during an FTP session can be controlled by using the **LISTFORMAT** operand of the SITE subcommand. For more information on the SITE subcommand, see “SITE” on page 93.

Unix-format List Fields

Unix-format responses returned by a z/VM FTP server contain the following fields, with each separated by one or more spaces:

mode a one byte *entry type* followed by three bytes each of Owner, Group, and Other *permissions*

link count
a count of the number of symbolic links to a file

owner the login or user name that owns a file or directory

group the group name with which permissions are associated

size the size (in bytes) of a file or directory

date date of last modification (month and day, provided in the form *mmm nn*)

time time of last modification, provided in the form *hh:mm* (for files greater than six months old, *hh:mm* is replaced with the year in which the file was last modified, provided as *nnnn*)

name the name of a file or directory

An example Unix-format response follows:

```
dir
>>>PORT 129,34,128,246,13,134
200 Port request OK
>>>LIST
125 List started OK
-rwx---r-x   1 MIKE     TCPIPDEV      240 Mar 18 16:13 LASTING.GLOBALV
-rwx---r-x   1 MIKE     TCPIPDEV     3192 Dec  8 1991  PROFILE.EXEC
-rwx---r-x   1 MIKE     TCPIPDEV        3 Feb  4 14:57 TELL.LOCK
-rwx---r-x   1 MIKE     TCPIPDEV      432 Mar 17 10:30 TEST.EXEC
-rwx---r-x   1 MIKE     TCPIPDEV       80 Mar 17  8:57 TEST.TEST
-rwx---r-x   1 MIKE     TCPIPDEV      432 Mar 18 10:28 TEST1.EXEC
-rwx---r-x   1 MIKE     TCPIPDEV    10640 Mar 18 13:28 TEST1.INFO
-rwx---r-x   1 MIKE     TCPIPDEV       80 Mar  8 17:37 TET.TET
-rwx---r-x   1 MIKE     TCPIPDEV       80 Mar  8 17:48 TET1.TET
-rwx---r-x   1 MIKE     TCPIPDEV       80 Mar  8 17:49 TET2.TET
-rwx---r-x   1 MIKE     TCPIPDEV    10640 Feb 26 16:36 T1.INFO
250 List completed successfully
Command:
```

Here is a sample Unix-format response for a virtual reader:

```
Command:
dir
>>>PORT 9,117,32,29,10,213
200 Port request OK.
>>>LIST
125 List started OK
-rwx---rwx   1 MIKE     -              0 Nov  3 12:46 0013.CIBULAPR.MAIL
-rwx---rwx   1 MIKE     -              0 Dec 29 10:12 0154.FL3XSAMP.TXT
-rwx---rwx   1 MIKE     -              0 Dec 16 11:39 0116.TCP-HELP.LIST3820
-rwx---rwx   1 MIKE     -              0 Dec 16 11:39 0115.TCP-OVER.LIST3820
-rwx---rwx   1 MIKE     -              0 Jun 29 1999  0153.FL32SAMP.TXT
-rwx---rwx   1 MIKE     -              0 Jan  1 12:50 0214.CIBULAMA.MAIL
250 List completed successfully.
Command:
```

Since the z/VM Byte File System (**BFS**) is based on a Unix file system model, all fields of the Unix-format file list are applicable and available when BFS files and directories are listed in response to a DIR subcommand.

When a Unix-format response is returned for z/VM-specific file system groups — **minidisks**, **SFS** directories, or virtual readers (**RDR**) — some of the information associated with this format is not pertinent to these file system groups. Thus, the FTP server substitutes or modifies values for certain Unix-format fields when its response corresponds to one of these file system groups; this is done to allow for correct processing of the response by web browser and graphical FTP clients.

Modifications to Unix-format field values for non-BFS files and directories are applied by the z/VM FTP server as follows:

- An *entry type* of **d** is supplied if the response entry is associated with an SFS directory; otherwise, a hyphen (**-**) is present.
- For SFS and minidisk, owner permissions are always **rwx**, but for RDR files, owner permissions are always hyphens (**---**).
- Group permissions are supplied as hyphens (**---**).
- Other permission bits will indicate the authority of the current user. Execute authority (**x**) is assumed if the login user has read authority for the file in use.

  For resources protected by an External Security Manager (ESM), Other permissions are always provided as **rwx**. Exact file authorizations for these resources may be determined by using the QAUTH operand of the "SITE" subcommand, see "SITE" on page 93.
- A link count of **1** is always provided.
- If the file owner is a member of an Access Control Interface group (ACIGROUP), that ACIGROUP name is supplied in the **group** field. If an ACIGROUP is not available, the group name is supplied as a hyphen (**-**). For more ACIGROUP information, see *z/VM: CP Planning and Administration*.
- For minidisk and SFS files, the **size** field contains the file size calculated depending on the planned mode. Because the final size depends on the transfer mode (ASCII, BINARY, or EBCDIC), the planned mode can be set with the LISTFORMAT command by specifying LISTFORMAT UNIX [ASCII | BINARY | EBCDIC].
- For virtual reader files, a size of **0** is always indicated.
- For a virtual reader, the spool ID precedes the file name and file type; thus, for these resources, this field is formatted as: *spoolid.filename.filetype*
- For SFS, a broken alias that points to an SFS file that is already deleted is shown with a file size of **0** and a date of 01/01/1970.

Transferring Files Using a Web Browser

To FTP to a z/VM host using a web browser, use the following FTP address or location format:

```
ftp://userid:password@host.domain/directory
```

If a *password* is not specified as part of the location information you supply (but is required to gain access to the host in question) a password prompt will be issued.

If a *directory* is not specified as part of the location information, the web browser informs the z/VM FTP server to begin at the root directory that is associated with the *userid* default working directory. If the user ID in question (*userid*) does not have authorization for the root directory, either proper authorization must be obtained or a suitable directory must be specified as part of the FTP location address (that is, a directory for which access authorization exists).

Also, be aware that the following sequence of events may produce unexpected results when a web browser FTP client is used in conjunction with a z/VM FTP server:

1. FTP operations commence with a root directory that corresponds to a specific type of z/VM file system (BFS, SFS, minidisk, reader) for which no specific directory has been specified as part of the FTP address or location.
2. The type of z/VM file system used is overtly changed, by having specified a fully-qualified directory in the FTP address or location.

3. A browser operation is performed that requires use of the root directory (and z/VM file system) that was originally established in Step 1, such as backward paging that initiates a file retrieval, store, or delete request.

Problems may arise after such a sequence of events because web browser implementations generally expect to operate with only one type of file system for a given user FTP session; the browsers do not expect the root directory to change for the life of an FTP session. Given this, a browser will not likely send a change directory (CD or CWD) request prior to an operation that requires a z/VM file system type that differs from that currently in use. Because the z/VM FTP server employs a different root directory for each type of z/VM file system in use, its responses may differ from those expected by the browser.

Under these (and similar) circumstances, unexpected results may be a consequence of discrepancies between the z/VM FTP server and the browser in use, with respect to conventions that concern how files should be referenced using path and directory information, as well as perception as to what constitutes a "working directory".

Transferring Files Using Secure FTP

The FTP client may be configured to use secure control and data connections using the Transport Layer Security (TLS) protocol. RFC 4217 (Securing FTP with TLS) describes the protocol used by the FTP client. Please see this RFC for more detailed information.

By default, the FTP client does not use secure control or data connections. To configure the FTP client to automatically use secure control connections, use the SECURECONTROL statement in the FTP DATA configuration file. To configure the FTP client to automatically use secure data connections, use the SECUREDATA statement in the FTP DATA configuration file.

Note: In order to secure FTP data connections, the control connection associated with the FTP session must first be secured.

To override the FTP SECURECONTROL and SECUREDATA settings at FTP command invocation, use the SECURE or NOSECURE option on the FTP command. This will cause the FTP client to either use or not use secure connections.

Note: The SECURE and NOSECURE options affect both control and data connections.

Once in the FTP session, the control connection may be secured using the CPROTECT subcommand. When the CPROTECT subcommand is issued, a negotiation will take place between the FTP client and the FTP server to secure the control connection. Once the control connection is secure, control connection transmissions are confidentiality and integrity protected by encryption. At this point, you may set up secure data connections by using the PRIVATE subcommand. When the PRIVATE subcommand is issued, a negotiation will take place between the FTP client and the FTP server to secure data connections. Once secure data connections are set up, data connection transmissions are confidentiality and integrity protected by encryption.To return to using non-secure (clear) data connections, the CLEAR subcommand may be used. To return to using non-secure (clear) control connections, the CCC subcommand may be used.

When FTP connections are secured, the FTP client always verifies the correctness of the remote server certificate and any associated CA certificates. Such verification is performed whenever a secure control or secure data connection is attempted. If a certificate error is detected, the corresponding connection fails. These operations are the same as when the following FTP DATA configuration file statements are used for the respective control or data connection types:

- SECURECONTROL YES CERTFULLCHECK
- SECUREDATA YES CERTFULLCHECK

ACCT

Purpose

Before you can access files on a foreign host, some hosts require account information. Use the ACCT subcommand to send host-dependent account information.

Operands

account_information
Specifies the account information required by the foreign host.

Usage Notes

- Some TCP/IP systems require passwords to gain access to specific resources or for a specific kind of access, such as “read” or “write”. For such systems, you can use the ACCT subcommand to supply such passwords.
- If the foreign host is a VM system, the ACCT subcommand can be used to supply minidisk passwords. The password you supply is used by the VM FTP server when CP LINK commands are issued on your behalf.

  When access to a VM minidisk is requested and a LINK password is required, the FTP server attempts LINK commands in the following order:

  1. MR (Multiple Read) mode
  2. WR (Read/Write) mode
  3. RR (Read only) mode.

  The level of access provided is determined by the first LINK attempt that succeeds. Thus, the minidisk password you supply as the *account_information* parameter should provide a minidisk LINK corresponding to the highest level of access you desire. For many systems, a Multiple Read password is most appropriate.

For more information about the CP LINK command, see the *IBM z/VM: CP Command and Utility Reference*.

APPEND

Purpose

Use the APPEND subcommand to append a local file to a foreign file.

Operands

localfile
The name of the local file to be appended to a file that resides on a foreign host. For information about how to specify *localfile* see "File Name Formats" on page 46.

foreignfile
The foreign host file to which the local file is to be appended. If a foreign file of this name does not already exist, the file is created. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

Usage Notes

- To append to a file on the foreign host, you must have a defined working directory, and you must have write privileges to it. For more information, see the subcommands "ACCT" on page 58 and "CD or CWD" on page 62.
- When the foreign file has a fixed-record format, the file format and record length of the foreign file are always preserved. Records from the local file can be truncated or padded with blanks when necessary.
- When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved. For more information, please refer to "Controlling File Translation" on page 51.

ASCII

Purpose

Use the ASCII subcommand to change the transfer type to ASCII, and at the same time change the record format used to store local files.

Operands

The ASCII subcommand has no parameters.

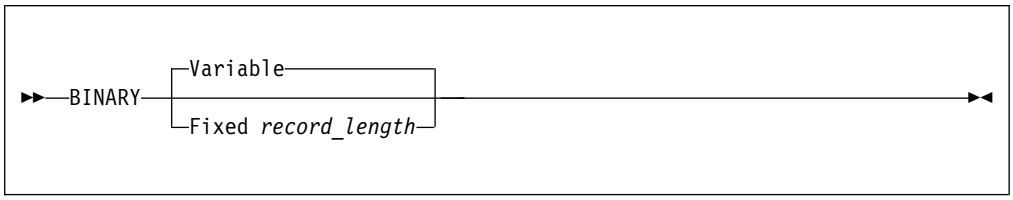
Usage Notes

- The ASCII subcommand is intended for use when text files are transferred to or from an ASCII host. When FTP sessions are established, ASCII is the default transfer type.

  Note: The ASCII subcommand causes files transferred to the local host to be stored as variable-record (**V**) format files.

For more information about file transfer methods, see Table 15 on page 52.

BINARY



Purpose

Use the BINARY subcommand to change the file transfer type to Image.

Operands

Variable
Specifies that files are stored as variable-record (**V**) format files.

Fixed *record_length*
Specifies that files are stored as fixed-record format (**F**), with records of length *record_length*.

Usage Notes

- To specify how a binary file is stored on the local host, use the BINARY subcommand with the VARIABLE or FIXED operand before a GET (or MGET) subcommand is issued to retrieve a file. Note that when the BINARY subcommand is used in this manner, this has the same effect as issuing separate TYPE IMAGE and LOCSITE subcommands.
- For a z/VM foreign host where the FTP server has been configured to perform automatic file translation, the BINARY subcommand has no effect on file translation — the server continues to determine whether EBCDIC-ASCII translation should occur based only on file extension. However, the VARIABLE and FIXED operands still affect how a transferred file is stored on the local VM system.

  To enable or disable automatic file translation for files transferred using the **Image** transfer type, use the AUTOTRANS operand of the SITE subcommand. For information, see “SITE” on page 93.

CCC

Purpose

Use the CCC subcommand to cause the FTP client to clear (stop using TLS on) the control connection. The control connection must already be in a secure state in order for the CCC subcommand to be accepted.

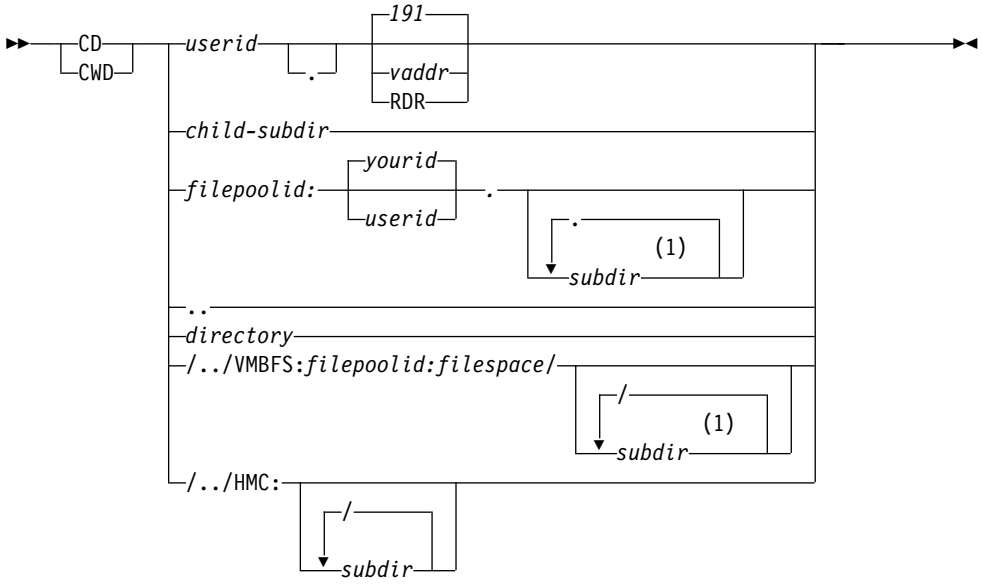
Operands

The CCC subcommand has no parameters.

Usage Notes

- If an error occurs while attempting to secure the control connection and the control connection enters an ambiguous state, the control connection will be aborted.
- If clearing the control connection is unsuccessful and the control connection is still in a secure state, the control connection will not be aborted
- The CCC option overrides an FTP DATA SECURECONTROL statement value of YES (described in FTP DATA File Statements section) and the use of the SECure option on the FTP command.
- Since clearing the control connection potentially allows an attacker to insert commands onto the control connection, some FTP servers may refuse to honor this command.

CD or CWD



Notes:

1 There is a limit to the name of a directory. See *z/VM: CMS Commands and Utilities Reference*.

Purpose

Use the CD or CWD subcommand to change the working directory or file group on the foreign host.

Operands

userid
The user ID associated with the resource that is to be accessed.

vaddr
A minidisk virtual address. The default virtual address is 191.

RDR
Indicates the virtual reader associated with *userid* is to be accessed.

child-subdir
A subdirectory that is defined under the current working directory.

filepoolid
The name of the Shared File System (SFS) file pool in which the directory to be accessed is defined.

userid
The user ID that owns the SFS directory that is to be accessed. If not specified, the user ID that corresponds to the currently active login user name (as specified with the USER subcommand) is used — signified here as *yourid*.

subdir
The name of a subdirectory that is defined under the top-level directory defined by *filepoolid:userid*.

`..` A special operand that changes the working directory to the parent directory on the foreign host. This operand performs the same function as the CDUP subcommand.

directory
The name of a file directory or other system-dependent file group designator on the foreign host.

/../VMBFS:*filepoolid:filespace/*
The name of the Byte File System (BFS) root. The **VMBFS**, *filepoolid*, and *filespace* tokens are all required and must be specified in **uppercase**.

/../HMC:
A keyword indicating a Hardware Management Console (HMC) removable medium (DVD or flash drive) that has been connected to the logical partition of a remote z/VM FTP server.

Usage Notes

- You can also use the CWD subcommand to change the current working directory. This subcommand is interchangeable with the CD subcommand.
- While in the FTP environment, you can issue a CD command to specify a subdirectory or fully-qualified directory name for access to SFS and BFS.
- To work with SFS directories, TCP/IP V2R3 for VM or later must be in use on the foreign host. To work with BFS directories, TCP/IP V2R4 or later must be in use on the foreign host. To work with VM readers, TCP/IP FL320 or later must be in use on the foreign host. To work with HMC directories, TCP/IP FL540 or later must be in use on the foreign host.
- When CD commands are used with a remote host running z/VM, your use of SFS, BFS, HMC, and virtual reader support can affect how the directory names you supply are used. When support for these resources is available, the VM FTP server will interpret the directory names you supply using the following precedence:
  1. If the current working directory is a BFS directory, the supplied directory name is treated as a BFS path name.
  2. If the string "`/../VMBFS:`" is present, the directory name is treated as a BFS path name.
  3. If the current working directory is an HMC directory, the supplied directory name is treated as an HMC directory name.
  4. If the string **/../HMC:** is present, the directory name is treated as an HMC directory name.
  5. If the current working directory is an SFS directory, the supplied directory name is treated as an SFS directory name.
  6. If a colon (:) is present in the directory name, it is treated as an SFS directory name.
  7. If the supplied directory name ends with either "` RDR`" or "`.RDR`", an attempt is made to change the working directory to a virtual reader.
  8. If the previously listed attempts fail to result in a successful change of directory, an attempt will be made to acquire a minidisk.

  Note: Certain SFS, HMC subdirectories, and BFS path names can present problems when you alternate the use of SFS/HMC/BFS resources and minidisks

or virtual readers as working directories. When SFS, HMC subdirectories, or BFS path names exist with names that also conform to the *userid.vaddr* or *userid*.RDR directory format, you might obtain that SFS, HMC, or BFS resource as a working directory instead of an intended *userid* minidisk or virtual reader. This situation is dependent upon the SFS, HMC, or BFS directory hierarchy being referenced, as well as the SFS/HMC/BFS directory that is the current working directory.

- The following restrictions apply to BFS files and directories:
  1. When a CD request is for a Byte File System (BFS) directory, only the VM user's primary GID (from the POSIXINFO statement) is used by the FTP server. The user's supplementary GID list (from the POSIXGLIST statement) is not used.
  2. When the user ID issuing a CD command has file pool administration authority for the target BFS file pool, that administration authority is not respected by the FTP server. In other words, the user ID must have explicit permission to the object through the UID and GID associated with the user ID.
- The initial working directory you obtain after you logon to a foreign host is dependent upon that host system. For z/VM hosts, the initial working directory established, as well as the type of access to that directory (with regard to "read" versus "write" status), can be affected by several factors, such as:
  – the logon user ID you supply
  – the configuration of the FTP server
  – the use of an External Security Manager (ESM)
  – the release of TCP/IP for VM in use on foreign host.

  In many environments, the 191 minidisk of the login user ID is established as the initial working directory, by default. When possible, write access to this resource is obtained. However, these defaults can be influenced and altered by one, or a combination, of the previously cited factors.
- If TCP/IP Function Level 320 or later is in use on the foreign host *and* native VM minidisk password protection is in use, access to minidisks will be achieved without the need to specify a minidisk password, when one of the following conditions is true:
  – the login user ID owns the target minidisk
  – a minidisk read, write, or multiple password of **ALL** is in effect for the minidisk

  For these conditions, write access will be obtained, whenever possible. If none of these conditions is applicable for this kind of environment, the **ACCT** subcommand must be used to provide a suitable minidisk password to gain access to the minidisk.

  For this same type of environment, but with a level of VM TCP/IP prior to Function Level 320 in use, minidisk access without the need for a minidisk password will be achieved *only* when a minidisk read, write, or multiple password of "ALL" is in effect for a minidisk. Otherwise, minidisk access will only be provided when you use the **ACCT** subcommand to provide a suitable minidisk password.
- You can specify the ***dev.null** directory for testing throughput. If *dev.null is specified, the PUT and MPUT subcommands transfer data to the foreign host, but do not store the data at the foreign host. The GET and MGET subcommands use the working directory that was in effect before the CD *dev.null command.

Examples

```
CD COOK.191
```

where 191 is the default minidisk address.

```
cd server5:.os2tools
```

where OS2TOOLS is the SFS directory within the client's filespace in file pool SERVER5.

```
cd /../VMBFS:SERVER5:ROOT/user/alan
```

where user/alan is the directory within file space ROOT of file pool SERVER5.

CDUP

Purpose

Use the CDUP subcommand to change the working directory to the parent directory on the foreign host. The CDUP command is a special case of the CD command. It is included to simplify the implementation of programs for transferring files between operating systems that use differing directory naming conventions. The reply codes are identical to the reply codes for the CD command. CDUP works for only one level at a time and is executed only if the current directory is an SFS or a BFS directory.

Note: You can also use the "**..**" operand of the CD subcommand to change the working directory to the parent directory.

Operands

The CDUP subcommand has no parameters.

CLEAR

Purpose

Use the CLEAR subcommand to cause the FTP client to set up for clear data connections when setup for secure data connections using TLS.

Operands

The CLEAR subcommand has no parameters.

Usage Notes

- CLEAR is not valid when there is no active security mechanism.

- CLEAR overrides an FTP DATA SECUREDATA statement value of YES and the use of the SECURE option on the FTP command.
- If set up for clear data connections is unsuccessful and the session is still setup for secure connections, secure data connections will continue to be used.

CLOSE

Purpose

Use the CLOSE subcommand to disconnect from the foreign host.

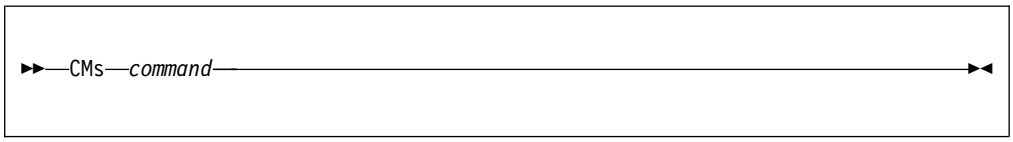
Operands

The CLOSE subcommand has no parameters.

Usage Notes

- CLOSE does not end FTP processing, therefore you can use the OPEN subcommand to establish a new session.

CMS



Purpose

Use the CMS subcommand to pass a CP or CMS command to the local z/VM system.

Operands

command
Specifies a CMS or CP command.

Usage Notes

- The CMS subcommand can be used to perform regular VM CMS functions such as checking your file list.
- Running TCP/IP related CMS functions may disrupt operation of the FTP command and cause its termination.

CPROTECT

Purpose

Use the CPROTECT subcommand to cause the FTP client to attempt to secure the control connection using TLS.

Operands

The CPROTECT subcommand has no parameters.

Usage Notes

- Once a CPROTECT subcommand succeeds, control connection transmissions are confidentiality and integrity protected by encryption.
- CPROTECT is not valid when there is no active security mechanism.
- The CPROTECT option overrides an FTP DATA SECURECONTROL statement value of NO and the use of the NOSECURE option on the FTP command.
- If an error occurs while attempting to secure the control connection and the control connection enters an ambiguous state, the control connection will be aborted.

DEBUG

Purpose

Use the DEBUG subcommand to enable or disable the internal debugging option.

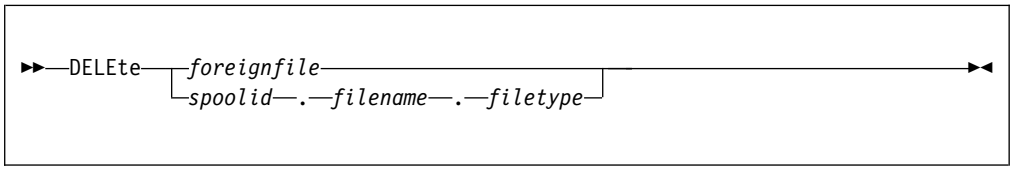
Operands

The DEBUG subcommand has no parameters.

Usage Notes

- DEBUG acts as a toggle that turns the debugging option on or off. The DEBUG subcommand is ON, initially, if the TRACE parameter was specified on the FTP command. The DEBUG subcommand is OFF if the TRACE parameter was not specified.
- The DEBUG subcommand is intended to aid in problem diagnosis. When DEBUG is ON, FTP displays tracing information, in addition to the commands sent to the foreign host and the responses received from that host.

DELETE



Purpose

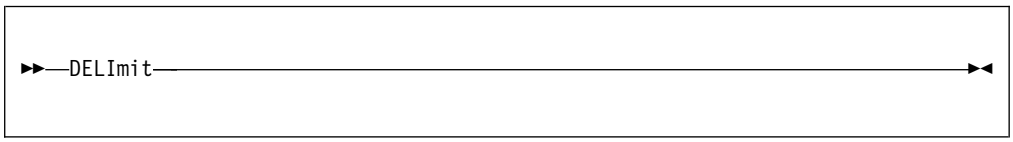
Use the DELETE subcommand to delete a file specified in the path name at the foreign host.

Operands

foreignfile
The foreign host file to be deleted. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

spoolid
The system-assigned number of the spool file to be deleted; *spoolid* is valid only when the working directory on the foreign host is a z/VM virtual reader (**RDR**). When a *spoolid* is specified, the *filename* and *filetype* associated with that spool file can optionally be included, though these values are not used by the FTP server.

DELIMIT



Purpose

Use the DELIMIT subcommand to display the character that is used as the delimiter between the *filename* and the *filetype*.

Operands

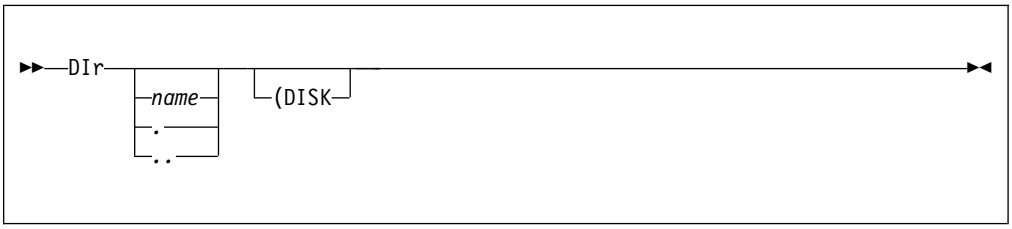
The DELIMIT subcommand has no parameters and should be used for informational purposes only.

Examples

- Response displayed after invoking the DELIMIT subcommand:

```
File delimiter is '.'
Command:
```

DIR



Purpose

Use the DIR subcommand to get a list of directory entries or a list of files in a file group on the foreign host.

Operands

name
The name of the directory or file group on the foreign host for which files should be listed. If *name* is omitted, all directory entries or files for the current working directory or file group are listed. For information about how to specify *name*, see the "Usage Notes" for this subcommand and "File Name Formats" on page 46.

To select which directory or file group is current, use the CD subcommand; for more information about this subcommand, see "CD or CWD" on page 62.

`.` Specifies that list information should be returned for the current working directory. For z/VM hosts, this operand is recognized in this manner only when SFS, HMC, or BFS directories are referenced. For other z/VM resources, responses are returned as if no operand had been specified.

`..` Specifies that list information should be returned for the *parent directory* of the current working directory. For z/VM hosts, this operand is recognized in this manner only when SFS, HMC, or BFS directories are referenced. For other z/VM resources, responses are returned as if no operand had been specified.

DISK
Stores the results of the DIR subcommand in file FTP DIROUTP, on the current local working directory. DIR subcommand results are not displayed when this operand is used.

Usage Notes

- For z/VM hosts that support list format selection, the response to the DIR subcommand can differ, based on the list format that is specified for the current session. If a list format of **UNIX** is specified, the server responds with a Unix-format list; otherwise, the response is a VM-format (**VM**) list.

  For more information about VM-format and Unix-format responses to the DIR subcommand, see "File List Formats".

  For more information about controlling the type of list format used for an FTP session, see the description of the "SITE" subcommand, refer to "SITE" on page 93.
- The DIR subcommand provides a complete list of directory and file entries that includes auxiliary information about those entries. To get a list that contains only the names of files present in the directory, use the LS subcommand. For more information, see "LS" on page 78.

- If the DIR subcommand is issued for a BFS directory that contains other than regular BFS files, a vm; foreign host may return an error response that indicates a BFS directory error has occurred.
- For z/VM foreign hosts, pattern matching can be used to specify a subset of files about which information is returned. For more information, see "File Name Pattern Matching" on page 47.
- z/VM hosts do not support pattern matching for BFS files and directories.
- If the working directory on the foreign host is a z/VM virtual reader (RDR) and VM-format lists are in use, pattern matching using the name operand is possible. When used, matching is performed against all reader-specific fields in unison (that is, data is returned for a reader file when a match is found for any field).

Examples

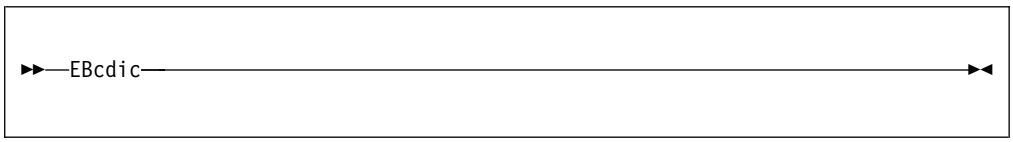
- A sample VM-format (**VM**) list response for a minidisk working directory follows:

```
dir
>>>PORT 129,34,128,246,13,134
200 Port request OK
>>>LIST
125 List started OK
ALTTEST  EXEC     V  36   12  1  2000-03-18  10:28:52  MKW191
LASTING  GLOBALV  V  48    5  1  2000-03-18  16:13:45  MKW191
PROFILE  EXEC     V  76   42  1  1998-12-08  13:44:42  MKW191
TEST     DATA     F  80    1  1  2000-03-17   8:57:33  MKW191
TEST     EXEC     V  36   12  1  2000-03-17  10:30:39  MKW191
TEST1    INFO     F  80  133  3  2000-03-18  13:28:39  MKW191
250 List completed successfully
Command:
```

- A sample Unix-format (**UNIX**) list response, for the same directory as in the previous example, follows:

```
dir
>>>PORT 129,34,128,246,13,134
200 Port request OK
>>>LIST
125 List started OK
-rwx---r-x   1 MIKEW    TCPIPDEV      432 Mar 18 10:28 ALTTEST.EXEC
-rwx---r-x   1 MIKEW    TCPIPDEV      240 Mar 18 16:13 LASTING.GLOBALV
-rwx---r-x   1 MIKEW    TCPIPDEV     3192 Dec  8 1998  PROFILE.EXEC
-rwx---r-x   1 MIKEW    TCPIPDEV       80 Mar 17  8:57 TEST.DATA
-rwx---r-x   1 MIKEW    TCPIPDEV      432 Mar 17 10:30 TEST.EXEC
-rwx---r-x   1 MIKEW    TCPIPDEV    10640 Mar 18 13:28 TEST1.INFO
250 List completed successfully
Command:
```

EBCDIC



Purpose

Use the EBCDIC subcommand to change the file transfer type to EBCDIC.

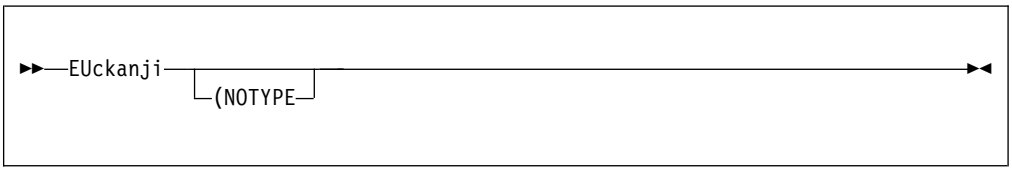
Operands

The EBCDIC subcommand has no parameters.

Usage Notes

- The EBCDIC transfer type is used to transfer files to or from another EBCDIC host. For more information about file transfer methods, see Table 15 on page 52.

EUCKANJI



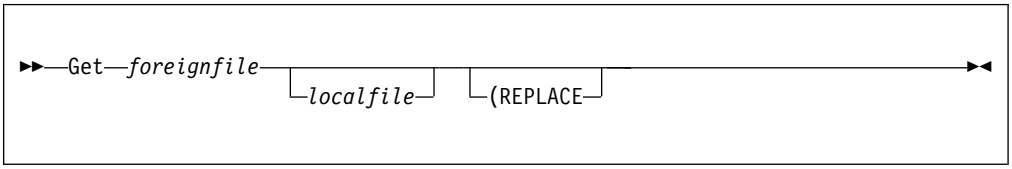
Purpose

Use the EUCKANJI subcommand to change the file transfer type to Extended UNIX Code (EUC) Kanji.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

GET



Purpose

Use the GET subcommand to transfer a file from a foreign host to the local host.

Operands

foreignfile
The foreign host file to be retrieved. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

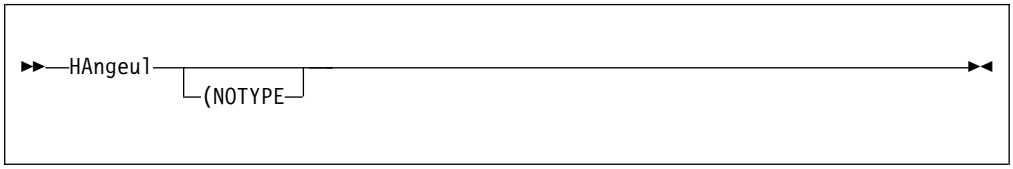
localfile
The name of the local file to be created. For information about how to specify *localfile*, see the "Usage Notes" for this subcommand and "File Name Formats" on page 46.

REPLACE
Causes any *localfile* that already exists to be overwritten with the content of the *foreignfile* that is retrieved. If *localfile* already exists and the REPLACE operand is not specified, FTP does not overwrite the existing file; it instead issues an informational message that identifies the existing file.

Usage Notes

- To retrieve a file from a foreign host, a working directory must be established for which you have at least "read" privilege. For more information, see the description of the subcommands "ACCT" on page 58 and "CD or CWD" on page 62.
- The *localfile* is created at the local working directory by default — that is, when a file mode is not specified or is specified as an asterisk (*).
- If *localfile* is not specified, FTP attempts to create a file that is named to match that of the retrieved foreign file (with measures taken to meet z/VM file naming conventions and restrictions). Thus, if *foreignfile* is a BFS file, *localfile* should be explicitly specified, since the default file name produced may not be as expected.
- If the foreign file is named using a Unix or Unix-like format, the *localfile* name and type are derived from that part of the *foreignfile* name that follows the right-most slash (/) that is present.
- The retrieval of a foreign file directly to a BFS directory is not supported by the CMS FTP client. However, this can be accomplished with a two-step process. First, use the FTP GET subcommand to retrieve the file to a local SFS directory or minidisk; then, use the CMS OPENVM PUTBFS command to copy that file to its final location.
- When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved.For more information, please refer to "Controlling File Translation" on page 51.

HANGEUL



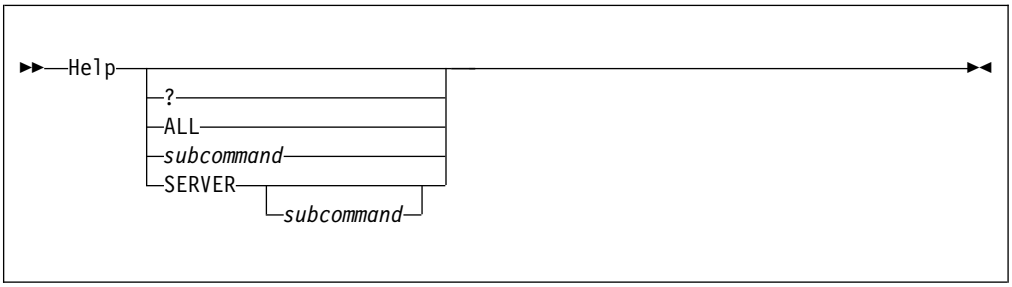
Purpose

Use the HANGEUL subcommand to change the file transfer type to Hangeul.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

HELP



Purpose

Use the HELP subcommand to get help with FTP subcommands.

Operands

? Provides information about how to use FTP.

ALL
Displays a description of all subcommands.

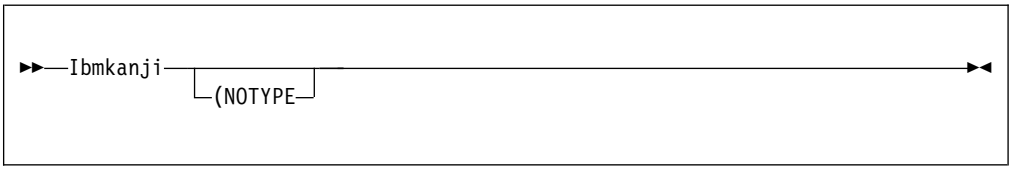
subcommand
Specifies the name of the subcommand. The *subcommand* can be abbreviated to its minimum abbreviation.

SERVER
Displays the help offered by the foreign host for the internal FTP commands.

Usage Notes

- If you enter the HELP subcommand without a parameter, you see the HELP FTP MENU, which lists the subcommands and a description of the help information available. If you enter the ? subcommand without a parameter, you see information about how to use the subcommands.

IBMKANJI



Purpose

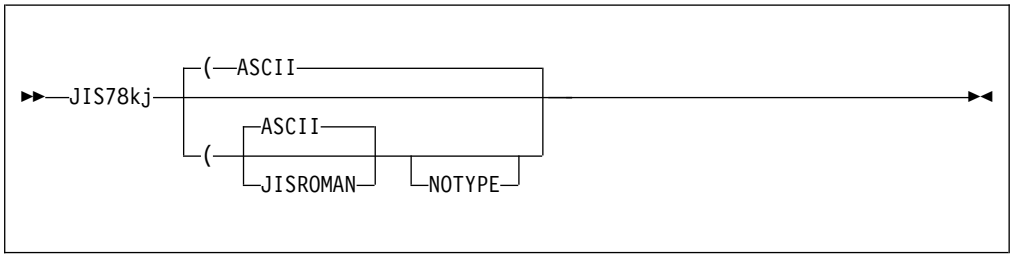
Use the IBMKANJI subcommand to change the file transfer type to IBM Kanji.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

This option usually causes *no* conversion to be performed on the transferred file. This option has exactly the same effect as the EBCDIC TYPE command alias.

JIS78KJ



Purpose

Use the JIS78KJ subcommand to change the file transfer type to JIS78KJ (1978 edition).

Operands

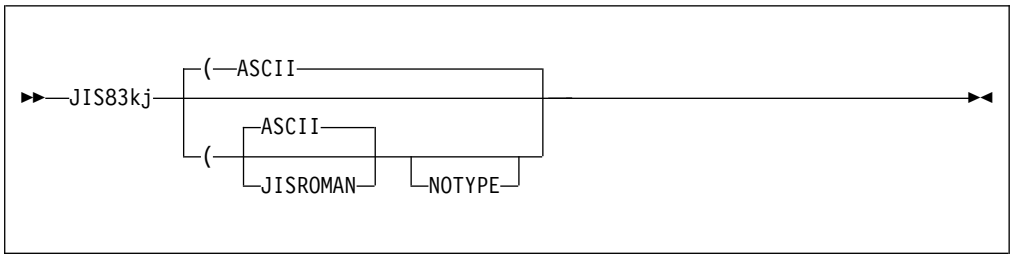
ASCII
Use ASCII shift-in escape sequence `ESC ( B`

JISROMAN
Use JISROMAN shift-in escape sequence `ESC ( J`

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

If neither ASCII nor JISROMAN is specified, then the ASCII shift-in sequence will be used.

JIS83KJ



Purpose

Use the JIS83KJ subcommand to change the file transfer type to JIS83KJ (1983 edition).

Operands

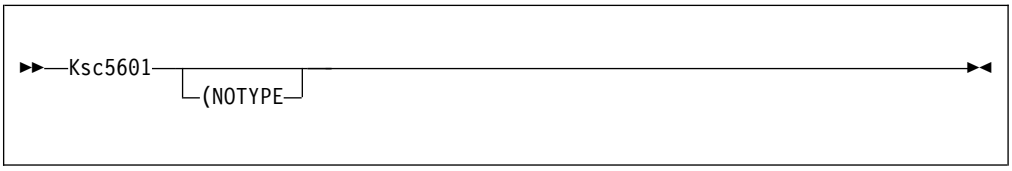
ASCII
Use ASCII shift-in escape sequence `ESC ( B`

JISROMAN
Use JISROMAN shift-in escape sequence `ESC ( J`

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

If neither ASCII nor JISROMAN is specified, then the ASCII shift-in sequence will be used.

KSC5601



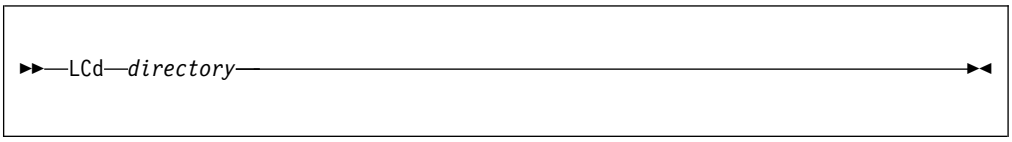
Purpose

Use the KSC5601 subcommand to change the file transfer type to KSC-5601.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

LCD



Purpose

Use the LCD subcommand to change the working directory on the local host.

Operands

directory
The CMS file mode of an accessed minidisk or SFS directory to be used as the current working directory on the local host.

Examples

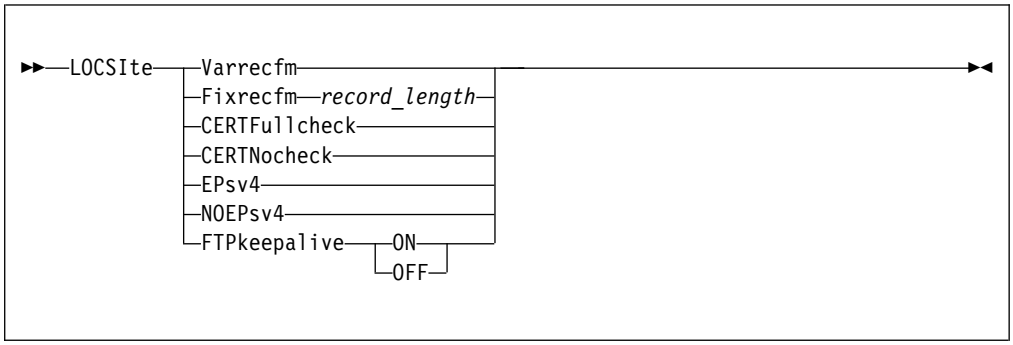
- Response displayed after invoking the LCD subcommand:

```
Local directory mode is 'Z'
Command:
```

Usage Notes

- LCD to a BFS directory is not supported by TCP/IP.

LOCSITE



Purpose

Use the LOCSITE subcommand to change the record format and record length used for files created on the local host, or to change how secure connections should be handled based certificate verification results, or to change how the connection should be established with the server for data transfer, or to set whether client will send keepalive packets for control connection periodically.

Operands

Varrecfm
Specifies that the variable record format is to be used.

Fixrecfm *record_length*
Specifies that the fixed-record format is to be used. The parameter *record_length* specifies the record length for fixed records.

CERTFULLCHECK
The remote server certificate and any associated CA certificates are verified for correctness, for control and data connections. If a certificate error is detected, the corresponding connection fails.

A self-signed certificate cannot be used for secure connections unless it has been added to the local certificate database. For more information, see Using Self-Signed Certificates in *z/VM: TCP/IP Planning and Customization*.

CERTNOCHECK
The CERTNOCHECK operand is accepted and ignored. All connections are handled as described for the CERTFULLCHECK operand.

EPsv4
Specifies that the FTP client will first try to use a EPSV or EPRT command to establish the connection with the server for data transfer. If the server rejects the EPSV or EPRT command, the client will then try to establish the data connection using the PASV or PORT command.

Note: If the server rejects either the EPSV or the EPRT command during the session, the client won't send EPSV or EPRT to the server again, even when EPSV4 is specified.

NOEPsv4
Specifies that the FTP client will send the server a PASV or PORT command to establish the connection with the server for data transfer.

FTPkeepalive ON
Specifies that the FTP client should make use of the TCP/IP server's keepalive mechanism to avoid timing out an idle control connection. The frequency with which packets are sent is determined by the KEEPALIVEOPTIONS statement in the TCP/IP server's configuration file.

FTPkeepalive OFF
Specifies that the FTP client should allow idle control connections to time out.

LOCSTAT

Purpose

Use the LOCSTAT subcommand to display local status information.

Operands

The LOCSTAT subcommand has no parameters.

The following status information is displayed:

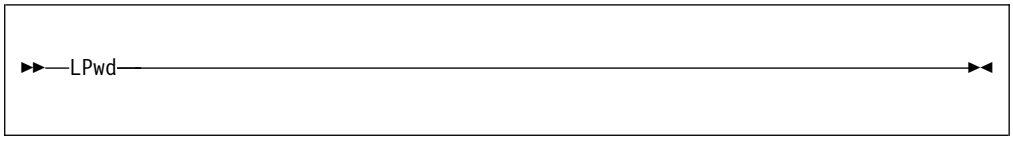
- Value of the TRACE flag (set using the FTP command or the DEBUG subcommand)
- SENDPORT setting (true or false)
- SENDSITE setting (true or false)
- Name, port number, and logon status of the foreign host
- Port number of the local host
- FTP data type (ASCII, EBCDIC, or Image) and transfer mode (block or stream)
- Record format (fixed or variable) and record length (for fixed record format)
- Translate table used by the client
- NETRC DATA file usage setting (true or false)
- Logon prompt setting (true or false)
- Console Width setting
- Secure control connection setting
- Secure data connection setting

Examples

- Information displayed after invoking the LOCSTAT subcommand:

```
Trace:FALSE, Send Port:TRUE
Use NETRC DATA File:TRUE, Logon Prompting:TRUE
Send Site with Put command:TRUE
Connected to:YKTVMX, Port:FTP control (21), logged in
Local Port:3452
Data type:a, Transfer mode:s
Record format:V
Secure control connection: TRUE (CERTFULLCHECK)
Secure data connections: TRUE (CERTFULLCHECK)
Translate Table: STANDARD
User Specified Translate Table: POSIX
Console Width: 80
Command:
```

LPWD



Purpose

Use the LPWD subcommand to display the name of the current working directory on the local host.

Operands

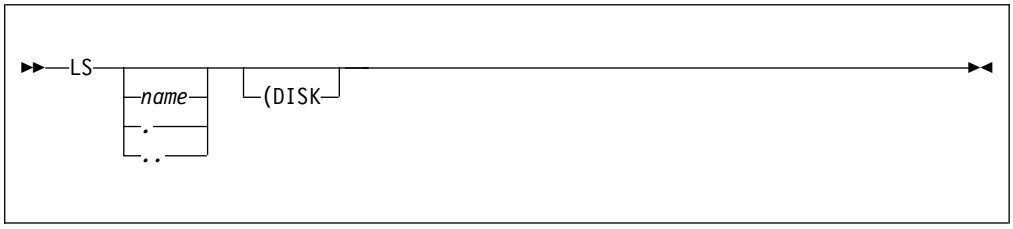
The LPWD subcommand has no parameters.

Examples

The following is an example of the response displayed as the result of the LPWD command.

```
Local directory is 'Z'
Command:
```

LS



Purpose

Use the LS subcommand to list only the name(s) of a set of foreign files, file group, or directory.

Operands

name
: The name of the directory or file group on the foreign host for which files should be listed. If name is omitted, all directory entries or files for the current working directory or file group are listed. For information about how to specify *name*, see the "Usage Notes" for this subcommand and "File Name Formats" on page 46.

  To select which directory or file group is current, use the CD subcommand. For more information see "CD or CWD" on page 62.

`.`
: Specifies that list information should be returned for the current working directory. For z/VM hosts, this operand is recognized in this manner only when SFS, HMC, or BFS directories are referenced. For other z/VM resources, responses are returned as if no operand had been specified.

`..`
: Specifies that list information should be returned for the *parent directory* of the current working directory. For z/VM hosts, this operand is recognized in this manner only when SFS, HMC, or BFS directories are referenced. For other z/VM resources, responses are returned as if no operand had been specified.

DISK
: Stores the results of the DIR subcommand in file FTP LSOUTPUT, on the current local working directory. DIR subcommand results are not displayed when this operand is used.

Usage Notes

- The LS subcommand provides a list of directory and file names only. To obtain a list of directory and file entries that includes auxiliary information about those entries, use the DIR subcommand. For more information, see "DIR" on page 69.
- For a given z/VM file system group, the response returned for an LS subcommand, regardless of whether VM-format or Unix-format lists are in use, is the same.
- If the LS subcommand is issued for a BFS directory that contains other than regular BFS files, a z/VM foreign host may return an error response that indicates a BFS directory error has occurred.
- For z/VM foreign hosts, pattern matching can be used to specify a subset of files about which information is returned. For more information see "File Name Pattern Matching" on page 47.

Note: z/VM hosts do not support pattern matching for BFS files and directories.

Examples

A sample response to an LS subcommand follows. In this example, pattern matching has been used to obtain a list of only those files for which the file name begins with a "T":

```
Command:
ls t*
>>>PORT 9,117,32,30,4,53
200 Port request OK.
>>>LIST
125 List started OK
TCPIP.DATA
TCPMNT2.NETLOG
TCPMNT2.SYNONYM
TCPSLVL.EXEC
TEST.EXEC
```

```
TESTFTPB.EXEC
TRACE2.TCPIP
250 List completed successfully.
Command:
```

A sample LS response for a z/VM virtual reader working directory is shown here:

```
Command:
ls
>>>PORT 9,117,32,29,10,213
200 Port request OK.
>>>LIST
125 List started OK
0013.CIBULAPR.MAIL
0154.FL3XSAMP.TXT
0116.TCP-HELP.LIST3820
0115.TCP-OVER.LIST3820
0153.FL32SAMP.TXT
0214.CIBULAMA.MAIL
250 List completed successfully.
Command:
```

MDELETE

Purpose

Use the MDELETE subcommand to delete multiple files on the foreign host.

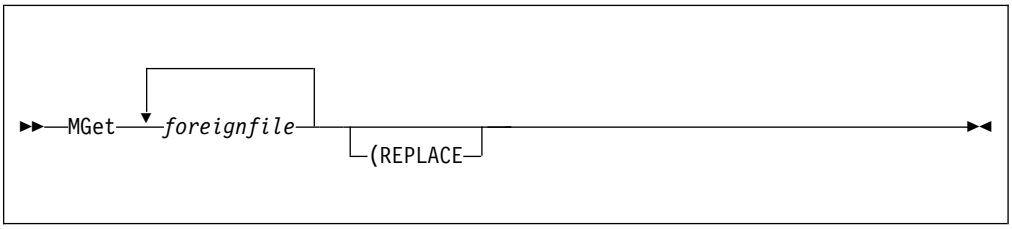
Operands

foreignfile
A foreign host file or file group to be deleted. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

Usage Notes

- For z/VM foreign hosts, pattern matching can be used to specify a subset of files to be deleted. For more information, see "File Name Pattern Matching" on page 47.
- To delete multiple files to which pattern matching cannot be applied, repeat *foreignfile* as necessary, with each *foreignfile* specification separated by at least one space.

MGET



Purpose

Use the MGET subcommand to transfer a file or group of files from a foreign host.

Operands

foreignfile
A foreign host file or file group to be retrieved. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

REPLACE
Causes an existing local file of the same name as *foreignfile* to be overwritten with the content of a corresponding *foreignfile* that is retrieved. If the local file already exists and the REPLACE operand is not specified, FTP does not overwrite the existing file; it instead issues an informational message that identifies the existing file.

Usage Notes

- To retrieve files from a foreign host, a working directory must be established for which you have at least "read" privilege. For more information, see the subcommands "ACCT" on page 58 and "CD or CWD" on page 62.
- For z/VM foreign hosts, pattern matching can be used to specify a subset of files to be retrieved. For more information, see "File Name Pattern Matching" on page 47.
- To retrieve multiple files to which pattern matching cannot be applied, repeat *foreignfile* as necessary, with each *foreignfile* specification separated by at least one space.
- The MGET subcommand creates local files in the same manner as those created (on a single-file basis) when the GET subcommand is used. For more information, see the GET subcommand "Usage Notes" on page 72.
- When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved.For more information, please refer to "Controlling File Translation" on page 51.

MKDIR

Purpose

Use the MKDIR subcommand to create a new directory on the foreign host.

Operands

directory
The name of the new directory to be created. The directory specified can be a fully-qualified SFS name, just an SFS directory name (without a specified filepool), a fully-qualified BFS name, or a BFS subdirectory name.

Usage Notes

- Because of the way FTP handles searches, avoid naming any SFS or BFS subdirectories with the same name as a CMS minidisk that you might want to access using FTP.
- MKDIR is allowed for the owner of the root directory where the new directory is to be created, or for SFS administrators and BFS super users. The POSIX and SFS permissions associated with the user ID you supply with the USER subcommand determine what directories can be created using the MKDIR command.

  General SFS users can create directories only in their own filespace. For BFS users, the POSIX permission checking performed by native BFS support determines whether the MKDIR can be done. An SFS file pool administrator or BFS super user has authorization to create directories for anyone enrolled in a filepool. The FTPSERVE machine is an SFS file pool administrator and a BFS super user, therefore inheriting all of the privileges of the registered user ID. See *z/VM: TCP/IP Planning and Customization* for more details.
- You must CD to an existing SFS or BFS directory before you can issue MKDIR.
- The VM FTP server initially creates an SFS file control directory. To change the SFS directory to Dircontrol, issue the SITE DIRATTR *dirid* DIRControl command. See the SITE command for more information.

For more information on SFS naming conventions, see the *z/VM: CMS User's Guide* or the *z/VM: SFS and CRR Planning, Administration, and Operation* book.

MODE

Purpose

Use the MODE subcommand to define how bits of data are to be transmitted, setting the mode (data format for the file transfer).

Operands

S Sets stream mode. Stream mode is the default transfer mode. In stream mode,

data is transmitted as a stream of bytes. Any representation type can be used with stream mode. Stream mode is more efficient, because data block information is not transferred.

B Sets block mode. In block mode, data is transmitted as a series of data blocks, preceded by one or more header bytes. Block mode allows the transfer of binary data and preserves the logical record boundaries of the file.

See Table 15 on page 52 for more information about file transfer methods.

MPUT

Purpose

Use the MPUT subcommand to transfer a file or group of files to a foreign host.

Operands

localfile
A local file or file group on the local host to be transferred to the foreign host. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

Usage Notes

- To transfer files to a foreign host, a working directory must be established for which you have at least "write" privilege. For more information see the subcommands "ACCT" on page 58 and "CD or CWD" on page 62.
- For z/VM foreign hosts, pattern matching can be used to specify a subset of local files to be transferred. For more information see "File Name Pattern Matching" on page 47.
- To transfer multiple files to which pattern matching cannot be applied, repeat *localfile* as necessary, with each *localfile* specification separated by at least one space or blank.
- The MPUT subcommand creates foreign files in the same manner as those created (on a single file basis) when the PUT subcommand is used. For more information see the Usage Notes for "PUT" on page 88.
- When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved. For more information, please refer to "Controlling File Translation" on page 51.

NETRC

Purpose

Use the NETRC subcommand to enable or disable automatic logon capability through use of a NETRC DATA file.

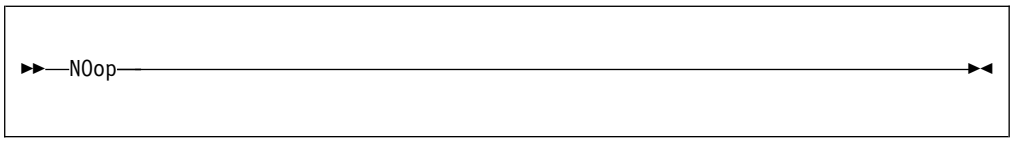
Operands

The NETRC subcommand has no parameters.

Usage Notes

- The NETRC subcommand acts as a toggle that enables or disables FTP's use of a NETRC DATA file. This file allows you to maintain logon user name and password information for specific hosts; when you connect to such a host, login is performed automatically, using the values defined in this file. For more information see "The NETRC DATA File" on page 31.

NOOP



Purpose

Use the NOOP subcommand to determine if the foreign host is still responding.

Operands

The NOOP subcommand has no parameters.

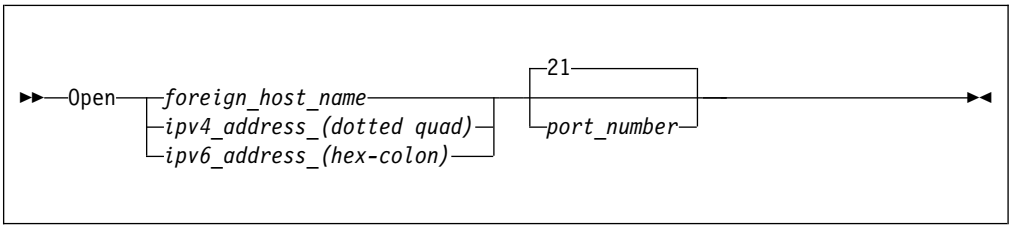
Usage Notes

- If the foreign host is responding, you receive one of the following responses.

  ```
  200 OK
  or
  200 NOOP command successful
  ```

  You are prompted for the next FTP subcommand after the message is displayed, unless the EXIT parameter of the FTP command is active.

OPEN



Purpose

Use the OPEN subcommand to open a connection to the foreign host's FTP server when:

- You want to open another connection after closing one without leaving the FTP subcommand environment
- You were unable to open a connection after specifying a *foreign_host* with the FTP command

Operands

foreign_host_name
The host name or internet address of the foreign host.

IPv4_address_(dotted quad)
"Old" style IP address in dotted quad notation. For example, "123.45.67.89".

IPv6_address_(hex-colon)
"New" style IP address in hex-colon notation. For example, "1234:5678::ABCD:EF09"

port_number
A port on the foreign host. The default is port 21, which is also known as a well-known port.

Usage Notes

- If you are already connected to a foreign host, you must disconnect from the foreign host before you can connect to a different host with the OPEN subcommand. For more information about closing a connection, see "CLOSE" on page 66.
- If a NETRC DATA file is in use and a valid match is found for the specified host name, that host's user name and password will be used to automatically logon to that host. If no match is found, or if the use of a NETRC DATA file is suppressed, automatic login will not occur; a subsequent USER subcommand must be issued in order to logon to the foreign host.

Examples

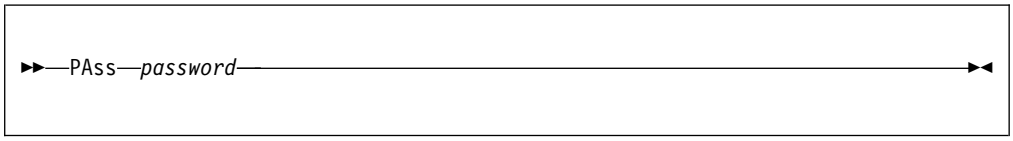
In this example, a CLOSE subcommand is used to terminate the FTP connection that is currently active, and an OPEN subcommand is then used to establish a connection with the GDLVMK4 host.

```
Command:
close
>>>QUIT
221 Quit command received.  Goodbye.
Command:
```

```
open gdlvmk4
Connecting to gdlvmk4 9.130.48.75, port 21
220-FTPSERVE IBM VM Level 420 at GDLVMK4.VMTEST.ENDICOTT.IBM.,
15:02:07 EST THURSDAY 2001-12-13
220 Connection will close if idle for more than 5 minutes.
USER (identify yourself to the host):
teri
>>>USER teri
331 Send password please.
Password:

>>>PASS ********
230-TERI logged in; working directory = TERI 191 (ReadOnly)
230 write access currently unavailable
Command:
```

PASS



Purpose

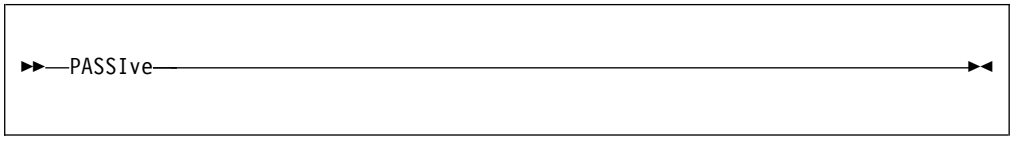
Use the PASS subcommand to supply a login password to a foreign host (if one is required by that host) in order to validate the identity of a previously supplied login user name.

Operands

password
The logon password to be used on the foreign host. If you supply an incorrect password, you are not prompted to enter the password again. You must instead issue a USER subcommand and again identify yourself to the remote host before you can supply the correct password using another PASS subcommand.

Note: If the password contains leading or trailing blanks, or begins with a single quote, it must be surrounded by single quotes and any imbedded single quotes must be doubled. Otherwise, the exact password should be entered. If a password containing leading or trailing blanks is entered without surrounding the password with single quotes, leading and trailing blanks will be removed from the password before sending the password to the FTP server.

PASSIVE



Purpose

Use the PASSIVE command to control whether the client or the server establishes connections for data transfers.

Operands

The PASSIVE subcommand has no parameters.

Usage Notes

- By default, passive data transfers are turned off when you start FTP. Each time you use the PASSIVE subcommand, passive data transfers are turned on and off alternately.If PASSIVE is on, the FTP client sends the server a PASV/EPSV command and uses the response to determine which address and port to use when establishing the data connection with the server. If PASSIVE is off, the client sends the server a PORT/EPRT command (if the SENDPORT subcommand is on) containing the address and port the server should use to establish the data connection with the client.
- Passive data transfer may enable the use of FTP through a firewall that does not allow incoming connections.

PRIVATE

Purpose

Use the PRIVATE subcommand to cause the FTP client to set up for secure data connections using TLS.

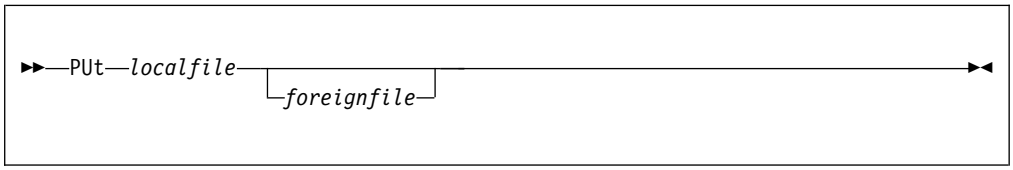
Operands

The PRIVATE subcommand has no parameters.

Usage Notes

- Once a PRIVATE subcommand succeeds, data transmissions are confidentiality and integrity protected by encryption.
- PRIVATE is not valid when there is no active security mechanism.
- PRIVATE overrides an FTP DATA SECUREDATA statement value of NO and the use of the NOSECURE option on the FTP command.
- PRIVATE is not allowed if the control connection is not secure
- If set up for secure data connections is unsuccessful, attempts to open a data connection will be rejected. To override this while in FTP session, enter the CLEAR command.

PUT



Purpose

Use the PUT subcommand to transfer a file from the local host to a foreign host.

Operands

localfile
: A file on the local host that is to be transferred to and created on the foreign host. For information about how to specify *localfile* see "File Name Formats" on page 46.

foreignfile
: The foreign host file to be created. For information about how to specify *foreignfile* see "File Name Formats" on page 46.

Usage Notes

- To transfer files to a foreign host, a working directory must be established for which you have at least "write" privilege. For more information, see the subcommands "ACCT" on page 58 and "CD or CWD" on page 62.
- If a file mode is not specified for the *localfile*, the file mode of the local working directory is first searched for this file; if found, this *localfile* is sent to the foreign host. If the *localfile* is not found at the local working directory, the first such file present in the CMS search order is transferred.
- In general, if a foreign host file already exists that matches the name of the transferred *localfile*, the existing foreign host file is replaced; this is the case for z/VM foreign hosts. For non-z/VM hosts, whether a foreign file is automatically replaced is dependent on the storage method used by that host.

  The "SUNIQUE" subcommand can be used to avoid overwriting a file that already exists on a foreign host. See "SUNIQUE" on page 98 for more information.
- Attempts to create the same foreign host file at a given time through different FTP connections may produce unexpected results. Some hosts may reply with a message that indicates a PUT or MPUT request should be resubmitted when such a condition is encountered.
- For z/VM hosts, when a transfer type of **Image** is used and the working directory on the foreign host is a minidisk or SFS directory, files are stored as variable-record (**V**) format files by default. To have files stored as fixed-record (**F**) format files for an Image mode transfer to these file system groups, do the following:
  1. Issue the SENDSITE subcommand to prevent SITE subcommands from being automatically issued for any PUT (or MPUT) subcommands that will be issued.
  2. Issue the SITE FIXRECM *nn* command to set the record length to be used when files are created.

- When an image file is stored on a z/VM foreign host as a fixed-record (**F**) format file, the last record of that file can be incomplete (that is, data stored using the last record does not consume the entire record). In such a case, this last record is padded with zeros (0).
- The transfer of a local file that resides in a BFS directory is not supported by the CMS FTP client. However, this can be accomplished with a two-step process. First, use the CMS OPENVM GETBFS command to copy that file from the BFS directory where it resides to a local SFS directory or minidisk; then, after this SFS directory or minidisk is established as the current local directory, issue the FTP PUT subcommand to transfer the file to the foreign host.
- When files are transferred between two hosts, appropriate transmission attributes must be used to ensure the content and structure of any transferred file are preserved.For more information, please refer to "Controlling File Translation" on page 51.

PWD

Purpose

Use the PWD subcommand to display the name of the current working directory on the foreign host.

Operands

The PWD subcommand has no parameters.

Examples

- The following is an example of the information displayed after invoking the PWD subcommand if the current working directory is a minidisk.

```
257 'KRASIK1.0191' is working directory
Command:
```

- The following is an example of the information displayed after invoking the PWD subcommand if the current working directory is an SFS directory.

```
257 'SERVER1:KRAS1' is working directory
Command:
```

- The following is a sample of the information displayed after invoking the PWD subcommand if the current working directory is a BFS directory.

```
257 './../VMBFS:BFS:KRASIK1/' is working directory
Command:
```

- The following is a sample of the information displayed after invoking the PWD subcommand if the current working directory is a VM reader.

```
257 "TERI.RDR" is working directory
Command:
```

QUIT

Purpose

Use the QUIT subcommand to disconnect from the foreign host and end FTP processing.

Operands

The QUIT subcommand has no parameters.

QUOTE

Purpose

Use the QUOTE subcommand to send an uninterpreted string of data to the server port on the foreign host. The QUOTE subcommand bypasses the FTP interface of the local user to send commands that the foreign server understands, but the local host does not understand.

Operands

string
Data to be sent verbatim to the port on the foreign host.

Usage Notes

- The QUOTE subcommand allows you to use commands that TCP/IP does not support, such as the FTP ALLOcate subcommand.

Examples

- To send the ALLOcate subcommand to a non-VM foreign host that supports ALLOcate, enter:

  ```
  QUOTE ALLO  bytes
  ```

  Where:

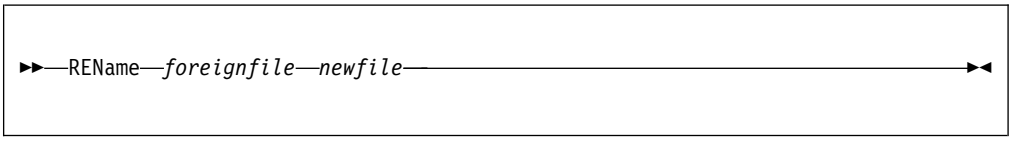
  ALLO Is the FTP standard string for the ALLOcate subcommand.

  bytes Is the number of bytes to allocate.
- To send a password to the VM FTP server, enter:

  ```
  QUOTE ACCT  password
  ```

In this example, the local host implements FTP in a way that does not support the ACCT subcommand, but the foreign host requires a password to obtain a write link to a minidisk.

RENAME



Purpose

Use the RENAME subcommand to rename a file on the foreign host.

Operands

foreignfile
The foreign host file that is to be renamed. For more information about how to specify *foreignfile* see "File Name Formats" on page 46.

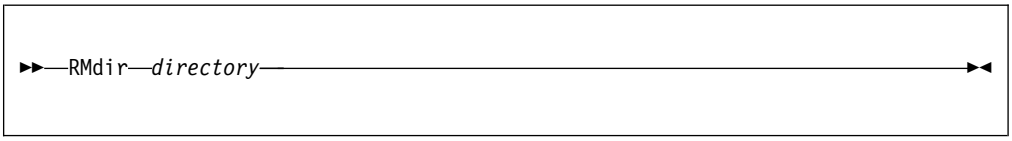
newfile
The new name to be given to *foreignfile* For more information about how to specify *newfile* see "File Name Formats" on page 46.

Usage Notes

- For a z/VM foreign host, you cannot use the RENAME subcommand to relocate a file in a different directory; a directory change is not permitted when a file is renamed.

  Similarly, you cannot replace an existing CMS file by using this subcommand. For example, if a file named *newfile* already exists, that file, as well as the existing *foreignfile* are maintained without change when a RENAME operation is attempted. In such a case, an error reply is returned that indicates the RENAME operation has failed.
- For a non-z/VM foreign host, the RENAME subcommand *may* cause an existing file to be deleted. That is, if the files *foreignfile* **and** *newfile* already exist on the foreign host, the content of the existing *newfile* file can be lost when its name is assigned to *foreignfile*.

RMDIR



Purpose

Use the RMDIR subcommand to remove a directory you own on a foreign host.

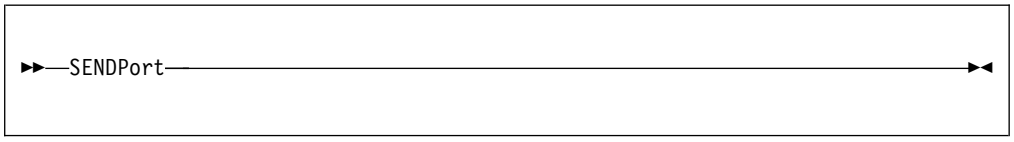
Operands

directory
The name of the directory to be removed.

Usage Notes

- Before issuing an RMDIR subcommand, the user ID identified in the USER subcommand must first be using SFS or BFS (have a working directory in some SFS or BFS directory).
- You cannot remove the current working directory. You must first change to a different directory, and then issue an RMDIR subcommand to remove the desired directory. Note that the usual restrictions regarding non-empty directories/subdirectories may prevent an RMDIR from being executed successfully. Refer to the *CMS User's Guide* for further information.
- The directory specified can be a fully-qualified SFS or BFS name or just an SFS or BFS directory name (without a specified file pool).

SENDPORT



Purpose

Use the SENDPORT subcommand to toggle PORT/EPRT commands.

Operands

The SENDPORT subcommand has no parameters.

Usage Notes

- By default, the SENDPORT subcommand is turned on when you start the machine. Each time you use the SENDPORT subcommand, it is turned alternately on and off.
- FTP uses a PORT/EPRT command by default when establishing a connection for each data transfer. FTP does not send PORT/EPRT commands for data transfer when you use the SENDPORT subcommand to disable PORT/EPRT commands.
- SENDPORT is useful for communication with FTP implementations that ignore PORT/EPRT commands, but show (incorrectly) that the PORT/EPRT command has been accepted.
- The SENDPORT setting is ignored when passive data transfer mode is established using the PASSIVE subcommand. This is because no PORT/EPRT command is transmitted by the client in passive mode.

SENDSITE

Purpose

Use the SENDSITE subcommand to toggle the sending of site information.

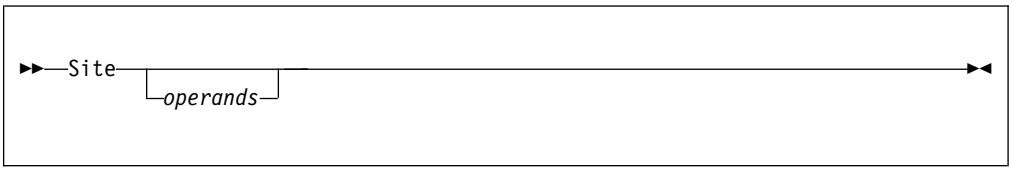
Operands

The SENDSITE subcommand has no parameters.

Usage Notes

- By default, the SENDSITE subcommand is turned on when you start FTP. Each time you use the SENDSITE subcommand, it is turned alternately on and off. If SENDSITE is on, site information is sent to the foreign host with the PUT or MPUT subcommands. If SENDSITE is off, no site information is sent.
- By default, FTP sends a SITE subcommand containing record format information, when you issue the PUT or MPUT subcommand. Record format information is used by VM and MVS™ foreign hosts.
- If the foreign host is non-VM, you can use the SENDSITE subcommand to prevent the record format information from being sent.

SITE



Note:

1. The recognition and validation of *operands* specified with the SITE subcommand is dependent on the foreign host.
2. The handling of incorrect or extraneous operand information may vary from one host to another. Some hosts may respond with an error reply code when a SITE command is considered to be in error, whereas another host may simply ignore such a command.

Purpose

Use the SITE subcommand to send information to a foreign host in order to make use of services which are specific to that system.

Operands

SITE subcommand operands that are supported by the *current* function level z/VM FTP server are described in this section. Some of these operands may not be supported by vm; hosts on which a prior function level is in use.

AUTOTrans ON | OFF
Specifies whether automatic EBCDIC-ASCII file translation, based on file extensions, is performed by a z/VM FTP server when files are transferred using the **Image** transfer type. Specify AUTOTRANS ON if automatic file translation should be performed, or AUTOTRANS OFF to prevent such translation. For more information about factors that affect file translation, see "Automatic File Translation" on page 52.

DIRATtr *dirid* **DIRControl | FILecontrol** *(options*
Sets the directory attribute of the SFS directory specified. For a further explanation of available options, see *z/VM: CMS User's Guide*.

FIXrecfm *record_length*
Specifies that the fixed-record format is to be used. The parameter *record_length* specifies the record length for fixed records.

GRAnt AUTh *fn ft dirid* **TO** *userid (options*
Grants authority for a user to a directory or files. *userid* can be a nickname in the FTP server's NAMES files. See the *z/VM: CMS Commands and Utilities Reference* for details of the GRANT AUTHORITY command.

LISTFormat VM | UNIX [ASCII | BINARY | EBCDIC]
Specifies the format to be used for file list information that is returned in response to DIR (or, LIST) subcommands. Specify LISTFORMAT VM for VM-format lists to be supplied by the FTP server, or LISTFORMAT UNIX if Unix-format lists should be returned. With LISTFORMAT UNIX, you can optionally specify the transfer mode (ASCII, BINARY, or EBCDIC) whose resulting size should be shown as file size in the output of the DIR (or, LIST) subcommands. The default transfer mode is ASCII. For more information about VM-format and Unix-format responses, see "File List Formats" on page 53.

PERMit *pathname mode_string (REPlace | ADD | REMove*
Issues an OPENVM PERMIT command to change the permission bits used to control the owner access, group access, and general access to a BFS object. *Pathname* can be specified as a relative pathname or a fully-qualified pathname. To issue the SITE PERMIT command, you must be the owner of the BFS object or have the appropriate privileges. See the *z/VM: OpenExtensions Commands Reference* for details of the OPENVM PERMIT command.

QAUTH *fn ft*
Queries the authority of the current working directory or files in the directory. The *fn* and *ft* parameters are optional. If *fn* or *ft* is omitted, QAUTH returns the directory authority information. See the *z/VM: CMS Commands and Utilities Reference* for details of the QUERY AUTHORITY command.

QDIRattr *dirid*
Returns the directory attribute of the SFS directory specified.

QDISK
Returns the disk information for the current working directory. If the current working directory is an SFS subdirectory, the information returned is the result of the CMS QUERY DISK and CMS QUERY LIMITS commands. If the current working directory is a BFS directory, the information returned is the result of

the CMS QUERY LIMITS command. See the *z/VM: CMS Commands and Utilities Reference* for details on the QUERY DISK and QUERY LIMITS commands.

QPERMit *pathname (OBJ*
Returns the output of the OPENVM LISTFILE (OWNERS command for the current working directory, or for the specified *pathname*. If *pathname* is omitted the QPERMIT command returns information for the current working directory. The *Pathname* can be specified as a relative pathname or it can be fully qualified. *OBJ* is optional and if specified, will return the output of OPENVM LISTFILE (OWNERS OBJECT command for the current working directory or pathname. See the *z/VM: OpenExtensions Commands Reference* for details of the OPENVM LISTFILE (OWNERS command.

REVoke AUTh *fn ft dirid* **FROM** *userid (options*
Revokes authority for a user to a directory or files. *userid* can be a nickname in the FTP server's NAMES files. See the *z/VM: CMS Commands and Utilities Reference* for details of the REVOKE AUTHORITY command.

TRANslate *filename*
XLATe *filename*
Specifies the name of the translation table to use on the server side. The translation table specifies the name of an EBCDIC-ASCII translation table that is to be used for all subsequent ASCII file transfers. To use the default translation table, issue either a SITE XLATE or a SITE XLATE * command. The specified translation table must be a TCPXLBIN file available on the remote z/VM or OS/390 host. See *z/VM: TCP/IP Planning and Customization* for further information on loading and customizing translation tables, as well as Chapter 15, “Using Translation Tables,” on page 437.

The name of translation table to use on the client side is controlled by the TRANslate option on the FTP command. See “FTP Command” on page 28.

VARrecfm
Specifies that the variable-record format is to be used.

Usage Notes

- To obtain information about site-specific operands that are supported by a foreign host, issue the **HELP SERVER SITE** command.
- By default, the z/VM PUT and MPUT subcommands automatically issue a SITE subcommand before a file is transferred. This is done to provide CMS file record format information to the foreign host (under the assumption that the foreign host can make use of this information, which is the case for IBM EBCDIC host systems).
- The SENDSITE subcommand can be used to prevent SITE subcommands from being automatically issued by the PUT or MPUT subcommands. For more information see “SENDSITE” on page 93.
- For the operands that follow, *dirid* can be specified as a single period (.) to signify the current foreign host working directory:

  DIRATTR, GRANT, QDIRATTR, REVOKE
- For SITE operands that require an SFS directory (*dirid*), specify this directory using the same syntax as for other SFS-related CMS commands. For more information about the naming of SFS directories, see the *z/VM: CMS Commands and Utilities Reference*.
- The operands that follow are accepted only when the foreign host working directory is an SFS directory:

DIRATTR, GRANT, QUATH, QDIRATTR, REVOKE

- The operands that follow are accepted only when the foreign host working directory is a BFS directory:

  PERMIT, QPERMIT

SIZE

Purpose

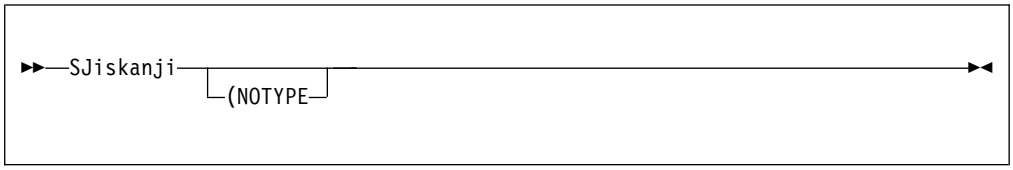
Use the SIZE subcommand to retrieve the transfer size (in bytes) for a foreign host file.

Operands

foreignfile
Specifies the foreign host file for which size information is to be retrieved. For more information about how to specify *foreignfile* see "File Name Formats" on page 46.

The returned file transfer size may account for included record delimiter or block header bytes, depending on the current TYPE and MODE settings.

SJISKANJI



Purpose

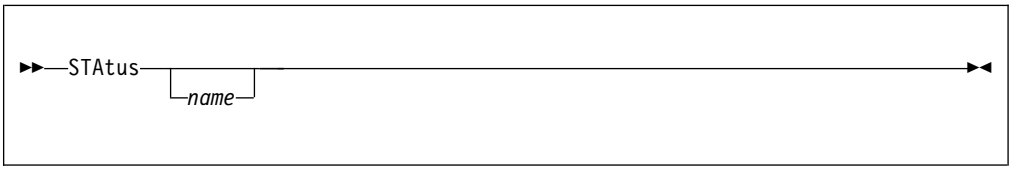
Use the SJISKANJI subcommand to change the file transfer type to SJISKANJI.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

Note: JIS is the abbreviation for the Japan Institute of Standards.

STATUS



Purpose

Use the STATUS subcommand to retrieve status information from the foreign host.

Operands

name
Specifies the file or foreign directory for which status information is requested. The *name* parameter is not supported by the VM FTP server.

Usage Notes

- The retrieved status information can be a directory, a file, or general status information, such as a summary of activity. If *name* is omitted, general status information is retrieved.

Examples

- Information displayed after invoking the STATUS subcommand:

```
211-Server FTP talking to host 129.34.128.246, port 3452
    User:  KRASIK1   Working directory:  KRASIK1 0191
    The control connection has transferred 399 bytes.
    There is no current data connection.
    The next data connection will be actively opened
    to host 129.34.128.246, port 3452, using
    mode Stream, structure File, type ASCII Nonprint,  byte-size 8.
    record format is V, record length 65535
    List format is UNIX. Automatic file translation is ON.
    FTPSERVE Translate Table: STANDARD
211 User Specified Translate Table: POSIX
Command:
```

STRUCT

Purpose

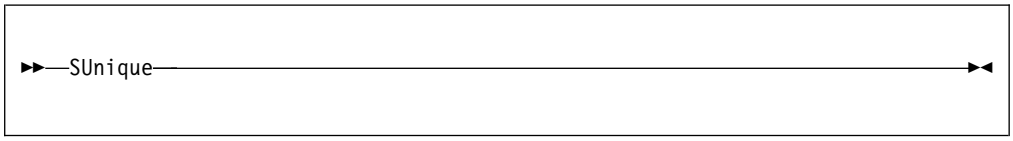
Use the STRUCT subcommand to set the structure of the file.

Operands

F Shows the file structure. The F file structure is the only structure supported.

The file structure affects the transfer mode, and the interpretation and storage of the file. With a file structure of F, the file is considered to be a continuous sequence of data bytes.

SUNIQUE



Purpose

Use the SUNIQUE subcommand to toggle the method of storing files.

Operands

The SUNIQUE subcommand has no parameters.

Usage Notes

- By default, SUNIQUE is toggled off, and FTP uses a store command (STOR) with the PUT and MPUT subcommands. If the foreign host already has a file with the name specified by *foreignfile*, the foreign host overwrites the existing file.
- If SUNIQUE is toggled on, FTP uses a store-unique command (STOU) with the PUT and MPUT subcommands, and prevents you from erasing the existing file on the foreign host that is specified by *foreignfile*. The created foreign file is stored with a unique file name. FTP sends the name of the created foreign file to the local host, where the file name is displayed on your terminal.
- To uniquely store files or copies of files when the named *foreignfile* already exists, the z/VM FTP server generates unique file names by appending a numeric suffix (from 1 to 999, in increments of 1) to the *foreignfile* name provided as part of a PUT or MPUT command. The *foreignfile* name is truncated (if necessary) to allow inclusion of this suffix. A maximum of 999 uniquely-named copies of a file can be created on a z/VM host. Note that the *foreignfile* name is modified only when necessary.

  When the "store unique" attribute is in effect, the FTP server identifies file names available for use by first searching the current working directory for the named file. If the file already exists, the server sequentially checks for similarly-named (and numbered) files until the next available suffix number for constructing a unique file name has been identified.

  Note:

  1. There is no guarantee that a file with a low suffix number (with respect to the numbering scheme just described) is an older file (with respect to time and date) than a file that has a higher suffix number.
  2. When uniquely-stored files are selected for deletion, some criteria other than the numeric suffix value should be used. The date and time stamp associated with a file is usually a reasonable compromise choice.

     For example, if a *foreignfile* name of UNIQTEST is in use, a file might be stored uniquely with the name UNIQTES9, with an ensuing file stored uniquely as UNIQTE10.

- In rare cases, the file transfers for which the "store unique" attribute is in effect may fail because the foreign host cannot uniquely store a file. For a z/VM host, this would be the case when the z/VM FTP server detects that the named file and all 999 uniquely named copies of the file already exist in the foreign directory. To store a file in this event, one of the following actions may prove successful:
  - Turn off the "store unique" attribute by re-issuing the SUNIQUE subcommand to toggle the setting, then re-attempt the file transfer.
  - Delete one of the uniquely named copies of the file you are attempting to store.

SYSTEM

Note: Information returned in response to the SYSTEM subcommand will vary depending on the foreign host with which an FTP session is established. Other factors, such as configuration parameters or active session attributes, may further affect this response. Note that the SYSTEM subcommand may not be supported by all foreign hosts.

Purpose

Use the SYSTEM subcommand to display information about the operating system that is in use on a foreign host.

Operands

The SYSTEM subcommand has no operands.

Examples

For z/VM hosts that support list format selection, the response to the SYSTEM subcommand can differ, based on the list format in effect for a session.

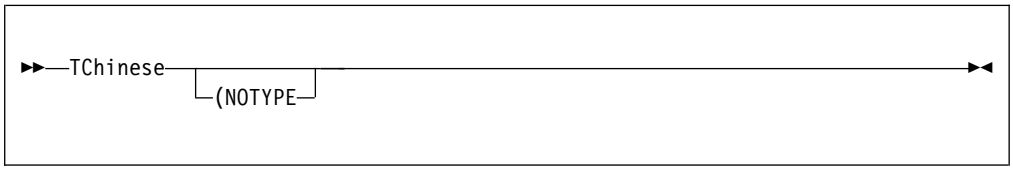
For example, when VM-format lists are in effect, the SYSTEM subcommand response is:

```
215-z/VM Version 3 Release 1.0, service level 0000 (nn-bit)
    CMS Level 16, Service Level 000
215 VM is the operating system of this server.
Command:
```

However, if Unix-format lists are in use, this response is modified to indicate this fact, and the following response is produced:

```
215-z/VM Version 3 Release 1.0, service level 0000 (nn-bit)
    CMS Level 16, Service Level 000
215 VM is the operating system of this server. UNIX list format is active.
Command:
```

TCHINESE



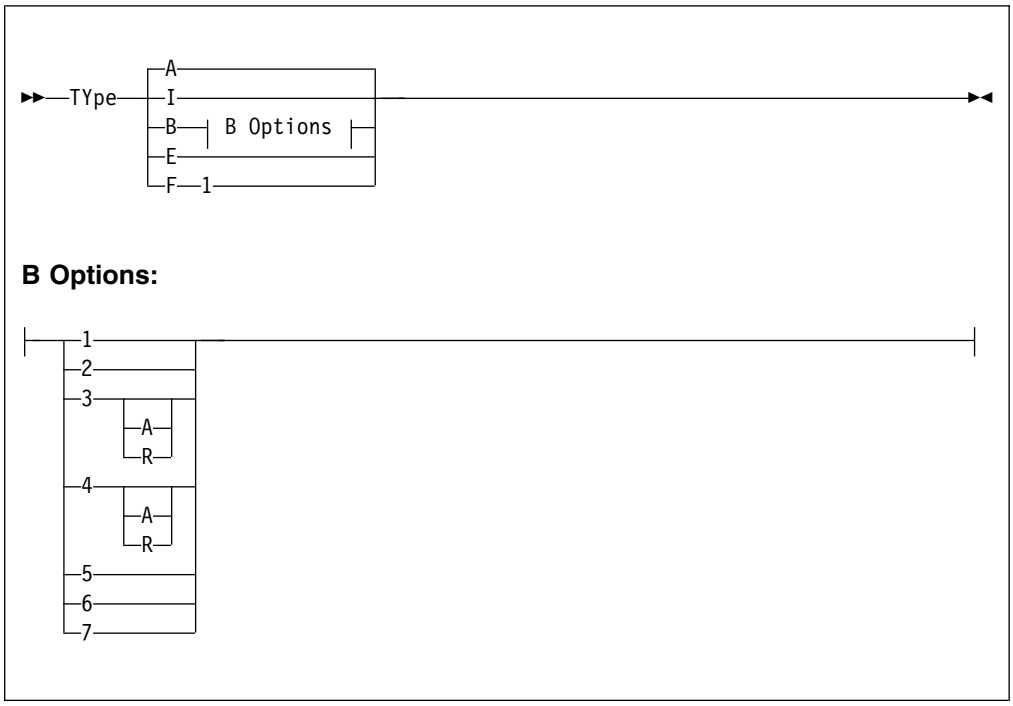
Purpose

Use the TCHINESE subcommand to change the file transfer type to Traditional Chinese.

Operands

NOTYPE
Used when connecting to an FTP server that does not support the TYPE command generated by this TYPE command alias. See Appendix D, "Using DBCS with FTP and Mail," on page 461 for more information.

TYPE



Purpose

Use the TYPE subcommand to set the file transfer type (the data representation for the transfer). FTP supports the ASCII, EBCDIC, Image (binary), and Kanji file transfer types.

Operands

A Sets the transfer type as **ASCII**, which is intended for use when text files are transferred to or from an ASCII host. When FTP sessions are established, ASCII is the default transfer type. The TYPE A subcommand has a similar effect as the ASCII subcommand, but it does not alter the record format used to store local files.

I Sets the transfer type as **Image** (or, **binary**), which is intended for use when binary data is transferred or when the efficient storage and retrieval of files is desired. With the Image transfer type, data is sent as contiguous bits, packed into 8-bit bytes. The TYPE I subcommand has a similar effect as the BINARY subcommand, but it does not alter the record format used to store local files.

Note: A transfer type of **Image** must be used when automatic file translation (performed by a z/VM FTP server) is desired.

B Sets the transfer type as DBCS Kanji, Hangeul, or Traditional Chinese. Specifying the B transfer type with the appropriate options has the same effect as using the EUCKANJI, HANGEUL, JIS78KJ, JIS83KJ, KSC5601, SJISKANJI or TCHINESE subcommands. The options provide further DBCS information. For more information about DBCS support for FTP, see Appendix D, "Using DBCS with FTP and Mail," on page 461.

E Sets the transfer type as EBCDIC. Specifying the EBCDIC transfer type has the same effect as using the EBCDIC subcommand. The EBCDIC transfer type is intended for efficient transfer between hosts that use EBCDIC for their internal character representation.

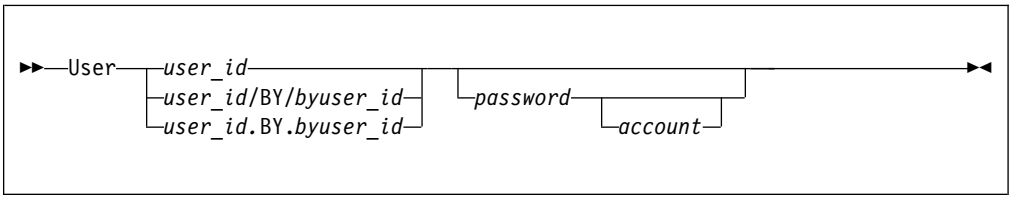
F Sets the transfer type as EBCDIC IBM Kanji. Specifying the IBM Kanji transfer type has the same effect as using the IBMKANJI subcommand. For more information about DBCS support for FTP, see Appendix D, "Using DBCS with FTP and Mail," on page 461.

Usage Notes

- For a z/VM foreign host where the FTP server has been configured to perform automatic file translation, the TYPE I subcommand has no effect on file translation — the server continues to determine whether EBCDIC-ASCII translation should occur based only on file extension.

  To enable or disable automatic file translation for files transferred using the **Image** transfer type, use the AUTOTRANS operand of the "SITE" subcommand. For more information, see "SITE" on page 93.
- For more information about file transfer methods, see Table 15 on page 52.

USER



Purpose

Use the USER subcommand to identify yourself to a foreign host after an FTP connection has been established.

Operands

user_id
The logon name to be used on the foreign host.

byuser_id
An alternate logon name *user_id* to be used by a foreign z/VM host when login authorization is performed.

When the /BY/ or .BY. delimiter and *byuser_id* are included with *user_id*, the *byuser_id* logon password is used for login authorization checking, instead of that for *user_id*. To be effective, *byuser_id* must be included in a LOGONBY statement in the *user_id* CP directory entry. When an External Security Manager (ESM) such as RACF® is installed, its defined authorization criteria may override that defined by CP for the LOGONBY statement. For example, RACF may perform authorization checks for attempts to log on a shared user ID. For further details, refer to the documentation provided by the ESM in use.

password
The logon password to be used on the foreign host. If a password is not included with the USER subcommand, you are prompted to provide a password, if one is required by the foreign host. If you supply an incorrect password, you are not prompted to enter the password again. You must instead reissue the USER subcommand to supply the correct password.

Note: If the password parameter contains blanks or begins with a single quotation mark, it must be surrounded by single quotation marks, and any imbedded single quotation marks must be doubled. Otherwise, the exact password should be entered. If a password containing blanks is entered without surrounding the password with single quotation marks, leading and trailing blanks will be removed from the password and only the first token of the password will be sent to the FTP server.

account
The host-dependent account information to gain access to specific resources or for a specific kind of access. If the account information contains blanks or begin with a single quotation mark, the account information must be supplied in the same manner described for the password above.

Using FTP Within an EXEC

When the FTP command is used within an exec, FTP subcommands can be supplied by either the program stack or a CMS disk file. Likewise, FTP command results can be directed to either the console, or to a CMS disk file.

Note:

1. FTP supports only DISK and TERMINAL I/O devices when FILEDEF commands are used to manage subcommand input and command results.
2. It is recommended that the EXIT option be used when FTP is used within an exec, so that error conditions can be detected and handled, as needed.

Providing FTP Subcommand Input

The program stack is used as the input source for FTP subcommands if the input device DISK has not been defined with a FILEDEF, or if the input device has been defined as TERMINAL.

If subcommand input is to be obtained from a disk file, that file must be identified with the CMS FILEDEF command. When disk file input is obtained by using FILEDEF, the following notes apply:

Note:

1. Only the ddname INPUT is recognized and used by FTP for subcommand input.
2. Program stack data is ignored.
3. If all FTP subcommands defined in the file complete without incident, and a QUIT was not the last subcommand executed, FTP will supply a QUIT subcommand to end the FTP session.
4. When a NETRC DATA file is used to provide logon user name and password values, it is advised that the NOPROMPT option be used to suppress unanticipated prompts for this information; this will cause FTP to return a nonzero return code in the event no NETRC DATA file exists or no entry for the target host is present in this file.
5. Because of the processing performed by FTP to obtain login passwords, you must provide the login password with the login name (user ID) when you use FTP within an exec, unless a NETRC DATA file is used to supply these values. (That is, for stack input, the logon password must be queued with the logon name. For disk file input, the password must be part of the same line as the logon name.)

   If a foreign host requires account information to be provided, you need to supply values with the ACCT subcommand in this manner as well.

Managing FTP Command Output

By default, FTP dialog output (messages, subcommand reply codes and text) generated during FTP command processing is directed to your terminal. However, FTP command results can be also directed to a disk file. As with subcommand input, the file to receive FTP dialog results must be identified with the CMS FILEDEF command. Only the ddname OUTPUT is recognized and used by FTP for handling FILEDEF command output.

Examples

The following example shows how FTP subcommands could be issued from within a REXX exec. In this example, the CMS program stack is used to supply FTP subcommands. This sample also illustrates a method for checking the return and reply codes from the FTP command.

```
/* FTPSTACK EXEC */
/*-------------------------------------------------------------*/
/* Setup variables to hold some commonly-used items.           */
/*-------------------------------------------------------------*/
remote_host = 'rdrunner.endicott.ibm.com'
login_id    = 'coyote'
pass_word   = 'wileyc'
acct_info   = 'secretpw'
local_mode  = 'A'
remote_dir  = 'COYOTE.191'
remote_file = 'ACME'     || '.' || 'SURPLUS'
local_file  = 'ACME-A1'  || '.' || 'CATALOG'
```

```
/*-------------------------------------------------------------*/
/* Setup the FTP subcommands to be issued...                   */
/*-------------------------------------------------------------*/
ftpcmd.1  = login_id pass_word  /* USER and PASS responses  */
ftpcmd.2  = 'acct' acct_info    /* Account info - this will */
                                /* supply the minidisk READ */
                                /* password in this case.   */
ftpcmd.3  = 'cd'  remote_dir
ftpcmd.4  = 'lcd' local_mode
/*-------------------------------------------------------------*/
/* Working with another VM host, so set up the transfer     */
/* MODE and TYPE for EBCIDIC data -- would do the same for  */
/* an MVS host.                                                */
/*-------------------------------------------------------------*/
ftpcmd.5  = 'mode b'
ftpcmd.6  = 'type e'
/*-------------------------------------------------------------*/
/* Get the file of interest, then quit.                        */
/*-------------------------------------------------------------*/
ftpcmd.7  = 'get' remote_file local_file
ftpcmd.8  = 'quit'
ftpcmd.0  = 8
/*-------------------------------------------------------------*/
/* Display and queue the FTP subcommands that will be used. */
/*-------------------------------------------------------------*/
'MAKEBUF'
buff_num = rc
Say "FTP subcommands being issued are:"
Do ii=1 to ftpcmd.0
  Say   ftpcmd.ii
  Queue ftpcmd.ii
End
/*-------------------------------------------------------------*/
/* Issue the FTP command, and save it's return code.        */
/*-------------------------------------------------------------*/
Say "" ; Say "Issuing FTP command..."
'FTP' remote_host '(EXIT'
ftprc = rc
'DROPBUF' buff_num
/*-------------------------------------------------------------*/
/* Interrogate the FTP command return code, and explain it  */
/* to the extent possible.                                     */
/*-------------------------------------------------------------*/
If (ftprc <> 0)
  Then Do
        fterr = Right(ftprc,3)
        cmdrc = Left(ftprc,(Length(ftprc)-3))
        Say "FTP command code:" cmdrc
        Say "FTP Server Reply code:" fterr
        Say "Possible Internal Error code:"  fterr
    End
Exit ftprc
```

This next example shows how FTP subcommands could be maintained separately from an exec that makes use of those subcommands. In this example, FTP subcommands to be issued are contained in a CMS file (SPACELY ACCTFTP A), which might contain the following data:

```
jetsong rorge
cd c:\sprocket\accts
lcd a
get sprocket.txt sprocket.acctdata
quit
```

For the data illustrated above, jetsong is the user ID and rorge is the login password. The GET subcommand will cause the file SPROCKET TXT A to be retrieved as SPROCKET ACCTDATA to the invoking user ID's A disk.

```
/* FTPFILDF EXEC */
/*-------------------------------------------------------------*/
/* Setup variables for host and input/output files.            */
/*-------------------------------------------------------------*/
remote_host = 'sprocketman.endicott.ibm.com'
input_file  = 'SPACELY ACCTFTP A'
output_file = 'SPACEFTP RESULTS A'
/*-------------------------------------------------------------*/
/* Check existing FILEDEFS, and account for them as            */
/* required.  Then establish the desired FILEDEFS.             */
/*-------------------------------------------------------------*/
'PIPE CMS QUERY FILEDEF | Stem filedefs.'
If (filedefs.0 = 0)
  Then Nop
  Else Nop           /* Add appropriate checking, etc. here */
'ESTATE' input_file
If (rc <> 0)
  Then Do
      Say input_file "does not exist..."
      Exit 8
    End
  Else 'FILEDEF INPUT DISK' input_file
'FILEDEF OUTPUT DISK' output_file
/*-------------------------------------------------------------*/
/* Issue the FTP command, and save it's return code.           */
/*-------------------------------------------------------------*/
Say "" ; Say "Issuing FTP command..."
'FTP' remote_host '(EXIT'
ftprc = rc
/*-------------------------------------------------------------*/
/* Interrogate the FTP command return code, and explain it     */
/* to the extent possible.                                     */
/*-------------------------------------------------------------*/
If (ftprc <> 0)
  Then Do
        fterr = Right(ftprc,3)
        cmdrc = Left(ftprc,(Length(ftprc)-3))
        Say "FTP command code:" cmdrc
        Say "FTP Server Reply code:" fterr
        Say "Possible Internal Error code:"  fterr
    End
/*-------------------------------------------------------------*/
/* Clear the FILEDEFS created for this FTP session.            */
/*-------------------------------------------------------------*/
'FILEDEF INPUT  CLEAR'
'FILEDEF OUTPUT CLEAR'
Exit ftprc
```

FTP Return Codes

When the FTP command EXIT operand is used, the FTP return code is composed of a command code and a reply code. The following is the format of the FTP EXIT return code:

YYXXX

where:

YY Is the command code, which is a number from 1 to 99. For a description of the possible FTP command codes, see Table 16 on page 106.

XXX Is the reply code that is sent from the foreign host FTP server. The reply

code is a 3-digit number. For a description of the possible reply codes, see Table 17 on page 108. At times *XXX* may instead be an FTP internal error code. For a description of possible internal error codes, see Table 18 on page 109. If an FTP server reply code or an internal error code is not available, *XXX* will be zero.

Table 16. FTP Command Codes

Code Number	Command	EXIT_IF_ERROR
1	AMBIGUOUS	true
2	?	false
3	ACCT	true
4	APPEND	true
5	ASCII	true
6	BINARY	true
7	CD	true
8	CLOSE	true
9	CMS	true
10	OPEN (CONNECT)	true
11	DEBUG	false
12	DELIMIT	false
13	DELETE	true
14	DIR	true
15	EBCDIC	true
16	GET	true
17	HELP	false
18	LOCSTAT	true
19	USER	true
20	LS	true
21	MDELETE	true
22	MGET	true
23	MODE	true
24	MPUT	true
25	NOOP	true
26	PASS	true
27	PUT	true
28	PWD	true
29	QUIT	true
30	QUOTE	true
31	RENAME	true
32	SENDPORT	true
33	SENDSITE	false
34	SITE	false
35	STATUS	true
36	STRUCT	true

Table 16. FTP Command Codes (continued)

Code Number	Command	EXIT_IF_ERROR
37	SUNIQUE	true
38	SYSTEM	true
40	TYPE	true
41	LCD	true
42	LOCSITE	true
43	LPWD	false
44	MKDIR	true
46	EUCKANJI	true
47	IBMKANJI	true
48	JIS78KJ	true
49	JIS83KJ	true
50	SJISKANJI	true
51	CDUP	true
52	RMDIR	true
53	HANGEUL	true
54	KSC5601	true
55	TCHINESE	true
56	NETRC	false
57	SIZE	true
60	PASSIVE	true
61	CPROTECT	true
63	PRIVATE	true
64	CLEAR	true
99	UNKNOWN	true

Examples

The following are examples of FTP return codes.

The FTP return code 16550 indicates the following:

16 The GET command failed.

550 The reply code from the FTP server.

The FTP return code 4532 indicates the following:

4 The APPEND command failed.

532 The reply code from the FTP server.

FTP Reply Codes

When you enter an FTP command, TCP/IP displays the sequence of subcommands, if any, that are sent to the foreign host's FTP server. In addition, the subcommand's response is also displayed as a reply code. These replies ensure the synchronization of requests and actions during file transfer, and guarantee that you

always know the state of the foreign host's FTP server. The descriptions of the possible reply codes are listed in Table 17.

Note: In general, the reply code descriptions below will not match the messages received from a foreign host, since reply message text will vary from one server implementation to another. The following descriptions are intended to provide an explanation of the reply codes themselves.

Table 17. FTP Reply Codes

Code	Description
110	Restart marker reply
120	Service ready in *nnn* minutes
125	Data connection already open; transfer starting
150	File status okay; about to open data connection
200	Command okay
202	Command not implemented; not used on this host
211	System status, or system help reply
212	Directory status
213	File status
214	Help message
215	VM is the operating system of this server
220	Service ready for new user
221	QUIT command received
226	Closing data connection; requested file action successful
227	The FTP server has opened a passive connection for data transfer at the specified IP address and port.
229	Entering extended passive mode.
230	User logged on; requested minidisk, BFS, or SFS Directory not available; proceed
234	Security data exchange complete
250	Requested file action or directory okay, completed
255	In target directory already
257	PATH NAME created or directory status given
331	Send password please
332	Supply minidisk password using account
421	Service not available; closing Telnet connection
425	Cannot open data connection
426	Connection closed; transfer ended abnormally
431	Temporarily unable to process security
450	Requested action not taken; file busy, minidisks or SFS directory not available
451	Requested action aborted; local error in processing
452	Requested action not taken; insufficient storage space in system
500	Syntax error; command unrecognized
501	Syntax error in parameters or arguments

Table 17. FTP Reply Codes (continued)

Code	Description
502	Command not implemented
503	Bad sequence of commands
504	Command not implemented for that parameter
521	Data connection cannot be opened with this PROT setting
522	Unsupported protocol. (1,2)
530	Not logged on
532	Need account for storing files
533	Command protection level denied for policy reasons
534	Request denied for policy reasons
550	Requested action not taken; file not found or no access
551	Requested action aborted; page type unknown
552	Requested file action ended abnormally; exceeded storage allocation
553	Requested action not taken; file name not allowed

Internal Error Codes

If an internal error occurs when you enter an FTP command, the reply code is an internal error code rather than an FTP reply code. Table 18 lists the possible internal error codes.

Table 18. Internal Error Codes

Code	Error	EXIT_IF_ERROR
01	NOmoreSLOTS	false
02	TOOfewDOTS	false
03	WOULDclobberFILE	false
04	CMSfileERROR	false
05	CMSfileNOTfound	false
06	CMSdiskFILEid	false
07	CMSinvalidCHAR	false
08	CMSdiskNOTaccessed	false
09	CMSdiskREADonly	false
10	CMSfileNOTaccessed	false
11	BLOCKbutNOTebcdcic	false
12	INITemulationERROR	true
13	OPENwasNOTissued	true
14	NOinputFILE	false
15	CANTwriteTOoutput	false
16	USERwasNOTissued	true
17	FileNOTauthorized	false
18	InvalidRecfm	false
19	InsuffStorage	false

Table 18. Internal Error Codes (continued)

Code	Error	EXIT_IF_ERROR
20	AppcVMerror	false
21	SharingConflict	false
22	SysResrcUnavail	false
23	FSOpenError	false
24	InvalidFILEDEFDev	false
25	NoPromptConflict	true
26	InvalidArgString	true
27	NoClosingQuote	true
28	TLSConnAuthFailed	EXIT_IF_ERROR
30	TLSConntrolNotAuth	true
31	TLSConnAlreadyAuth	true
32	TLSConnAlreadyClear	true

For example, the internal error code 13 indicates that an "`OPENwasNOTissued`" error condition has occurred.

Using FTP with RACF

The Resource Access Control Facility (RACF) allows FTP servers to act as *surrogates* for other user IDs. This means that the server can access those disks available to that user ID.

The command that allows FTP servers to act as surrogates is provided in a program called FTPPERM EXEC. To use it, enter the command:

```
FTPPERM ADD
```

If you get an error, contact your system administrator.

You may delete the FTP server's surrogate authority by issuing the command:

```
FTPPERM DELETE
```

FTPPERM is explained in *z/VM: TCP/IP Planning and Customization*.

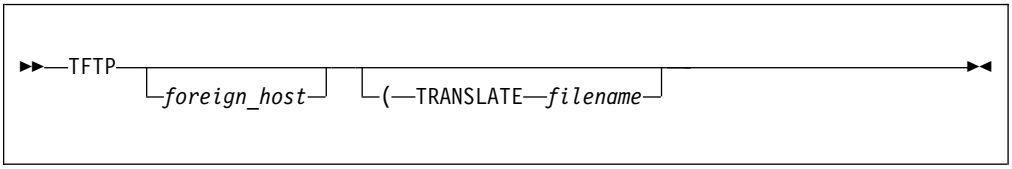
Chapter 3. Transferring Files Using TFTP

This chapter describes how to transfer files using the Trivial File Transfer Protocol (TFTP) command and its subcommands. The TFTP command is a simple method to get files from, and send files to, a foreign host.

TFTP uses the User Datagram Protocol (UDP), and uses a simple lock-step method to acknowledge each packet separately. TFTP cannot list directories and has no provision for user authentication.

You can also use the FTP command to transfer files from a VM host. For more information about the FTP command, see Chapter 2, "Transferring Files Using FTP," on page 27.

TFTP Command



Purpose

Use the TFTP command to transfer files from or to the foreign host.

Operands

foreign_host
Specifies the name of the foreign host to which you are connecting. Specify the foreign host by its host name or its internet address. You are prompted for a host name if you do not specify a foreign_host with the TFTP command. If you specify foreign_host incorrectly, or if the foreign host is not accessible, you enter the TFTP command shell. To identify the desired host, use the OPEN subcommand. See "OPEN" on page 115 for more information. Specify the foreign host by its internet host name:

internet-hostname

or by its dotted-decimal internet address:

nnn.nnn.nnn.nnn

TRANSLATE *filename*
Specifies the file name of a translation table file other than a standard table. The file type is TCPXLBIN, and the file mode is an asterisk (*). If this parameter is not specified, the TFTP command searches sequentially for the TFTP TCPXLBIN or STANDARD TCPXLBIN table files. If neither is found, TFTP uses the compiled translation table. TFTP TCPXLBIN is not supplied, because the standard translation table is adequate for most applications. You can create your own TCPXLBIN table file if your installation needs a different translation.

© Copyright IBM Corp. 1987, 2016

Creating Translation Tables

You can edit and modify translation table files that have a file type of TCPXLATE. A translation table must be in binary format and have a file type of TCPXLBIN. To convert a table from modifiable form to binary form, use the CONVXLAT EXEC file written in the REXX programming language.

Creating your own translation table provides the following advantages.

- You do not require the application's source code.
- You do not need to recompile the application to change the translation table.
- You can specify, as a command parameter, a translation table file that is different from that of your system administrator.

For information about creating your own translation tables, see *z/VM: TCP/IP Planning and Customization*.

TFTP Subcommands

The TFTP subcommands and their parameters are described in detail.

GET

Purpose

Use the GET subcommand to retrieve a file from the TFTP server.

Operands

foreignfile
Specifies the name of the file to be retrieved from the foreign host.

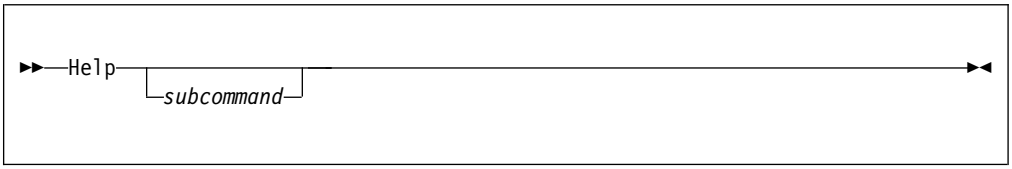
localfile
Specifies the name of the local file to be created. The *localfile* is specified by *filename.filetype.filemode* or *filename.filetype*. The default *filemode* is A.

Attention: If a file with the name specified by *localfile* already exists, that file is overwritten.

Usage Notes

1. The *localfile* is created or overwritten with a variable record format. The file is transferred using the currently selected transfer mode. If the transfer mode is OCTET, the file is created with a record length of 512.
2. When a file is stored in ASCII mode, each empty line is written to the file as a single space, because a record of zero length cannot be written to a CMS file. When a file is transferred to and from VM, each previously empty line contains a single space. See "MODE" on page 114 for more information about transfer modes.

HELP



Purpose

Use the HELP subcommand to receive assistance with the TFTP subcommands.

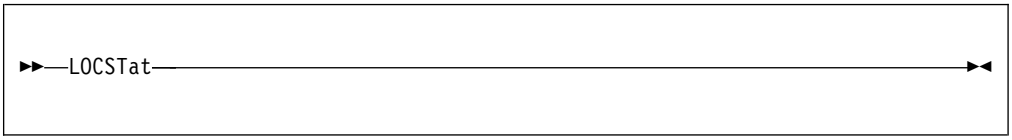
Operands

subcommand
Specifies the name of the TFTP subcommand.

Usage Notes

- If you enter HELP without a parameter, you see the list of TFTP subcommands that are supported by the client.

LOCSTAT



Purpose

Use the LOCSTAT subcommand to display the local status information about TFTP.

Operands

The LOCSTAT subcommand has no parameters.

Usage Notes

- The following local status information is displayed when the LOCSTAT subcommand is issued:
  - Name of connected host (or the message `not connected`)
  - Transfer mode
  - Packet tracing (on or off)
  - Packet retransmit interval, in seconds
  - Packet retry count, in packets

MAXPKT

Purpose

Use the MAXPKT subcommand to set the maximum number of times a packet is retransmitted.

Operands

number_of_times
Specifies the maximum number of times a packet is retransmitted. The default value is 5 times.

Usage Notes

1. When TFTP sends a packet for which a response packet is expected from the foreign TFTP server, a timer is set, as specified by the REXMIT command. See "REXMIT" on page 116 for more information. If the time interval expires before the response packet is received, the original packet is retransmitted.
2. When the packet has been retransmitted the maximum number of times (as specified by the MAXPKT command) and nothing is received from the foreign TFTP server, the transfer is terminated.
3. If the transmission error rate is high, you can increase the number to a value greater than 5.

MODE

Purpose

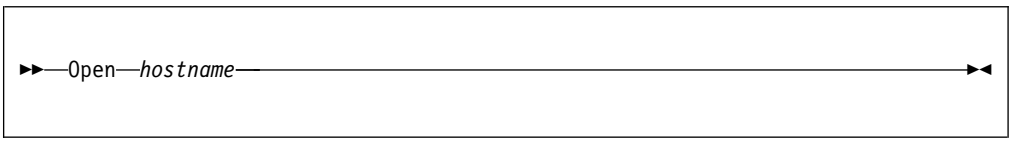
Use the MODE subcommand to change the transfer mode. The TFTP transfer mode determines how the data bits are transmitted.

Operands

NETASCII
Sets the mode to ASCII. In NETASCII mode, data is transferred in ASCII. This is the default.

OCTET
Sets the mode to Image (binary). In OCTET mode, all data is treated as raw 8-bit bytes. No conversion is performed on the bytes.

OPEN



Purpose

If you did not connect to a foreign host when you invoked the TFTP command, you must open a connection to a foreign host before you can transfer files. Use the OPEN subcommand to connect to the desired host.

Operands

hostname
Specifies the internet host to which you are transferring files. Specify the foreign host by its internet host name:

internet-hostname

or by its dotted-decimal internet address:

nnn.nnn.nnn.nnn

Usage Notes

- If the internet host name parameter of the TFTP command is not specified, you enter the TFTP command shell. To identify the desired host, use the OPEN subcommand. See "TFTP Command" on page 111 for more information.

PUT

Purpose

Use the PUT subcommand to send a file to the foreign TFTP server.

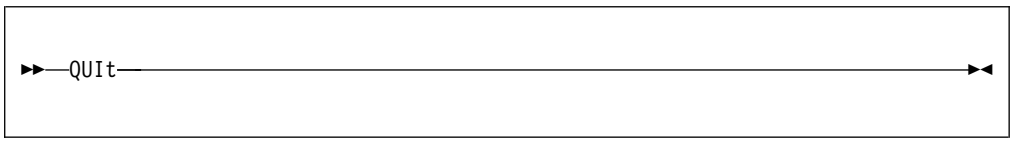
Operands

localfile
Specifies the name of the local file to be transferred to the foreign host. The *localfile* is specified by *filename.filetype.filemode* or by *filename.filetype*. The default *filemode* is A.

foreignfile
Specifies the name of the file to be created on the foreign host.

The file is transferred using the currently selected transfer mode. See "MODE" on page 114 for more information.

QUIT



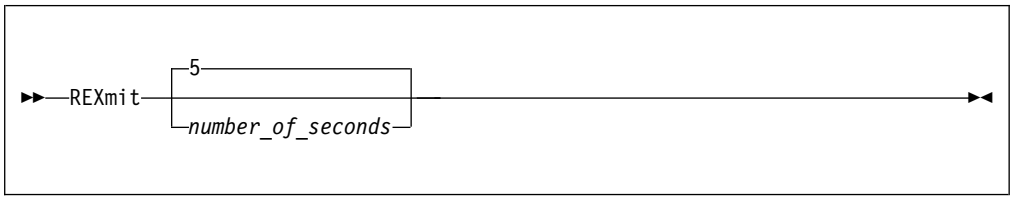
Purpose

Use the QUIT subcommand to end the TFTP command.

Operands

The QUIT subcommand has no parameters.

REXMIT



Purpose

Use the REXMIT subcommand to change the time-out value for each packet.

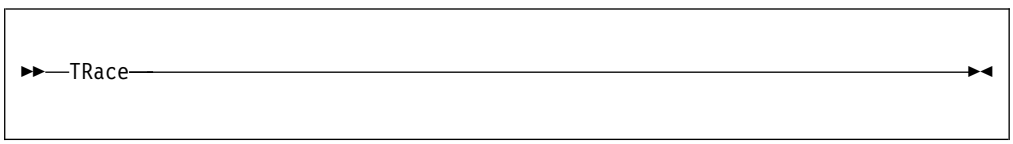
Operands

number_of_seconds
Specifies the number of seconds TFTP waits before retransmitting a packet. The default value is 5 seconds.

Usage Notes

- When TFTP sends a packet for which it expects a response packet from the foreign TFTP server, a timer is set, as specified by the REXMIT command. If the time interval expires before the response packet is received, the original packet is retransmitted.
- When the packet has been retransmitted the maximum number of times (as specified by the MAXPKT command) and nothing is received from the foreign TFTP server, the transfer is terminated. See “MAXPKT” on page 114 for more information.
- You can increase the *number_of_seconds* value for a connection with a slow transmission rate.

TRACE



Purpose

Use the TRACE subcommand to toggle the tracing of TFTP packets.

Operands

The TRACE subcommand has no parameters.

Usage Notes

- When TRACE is enabled, information about each TFTP packet that is sent or received is displayed. The following information about each packet is displayed.

  Field **Description**

  direction
  Indicates either *Sending* or *Received*.

  (size) Specifies the number of bytes in the packet.

  kind Specifies the type of TFTP packet. The TFTP packet types are:
  RRQ Read request
  WRQ Write request
  Data Data packet
  ACK Acknowledgment packet
  Error Error packet

  per-packet-information
  Contains other data contained in the packet. The type of information displayed about each packet is:
  RRQ Foreign file name, transfer mode
  WRQ Foreign file name, transfer mode
  Data Block number
  ACK Block number
  Error Error number, error text (if any)

Chapter 4. Sending and Receiving Electronic Mail

This chapter describes electronic note and file delivery and also discusses how mail is sent to and from other hosts/users on systems connected through TCP/IP or Remote Spooling Communication Subsystem (RSCS). In addition, this chapter also describes the electronic mail gateway, delivery failures, the OfficeVision interface, IMAP, and the SMSG interface to the SMTP server.

You can also use SMTP to transfer Kanji mail messages. For more information about using Kanji support with SMTP, see "Using DBCS with Mail" on page 465.

NOTE and SENDFILE Commands

Note: The SENDFILE and NOTE commands are no longer supplied with TCP/IP. You can now use the CMS version of SENDFILE and NOTE to send notes and files to network recipients. For a complete description of the syntax and options for these commands, refer to the *z/VM: CMS Commands and Utilities Reference*.

Electronic Mail Gateway

Electronic mail services are provided in conjunction with the SMTP virtual machine. This virtual machine can be configured by your TCP/IP administrator to operate as a mail gateway between TCP/IP network users and users located on an RSCS network that is attached to the local host. For example, PROFS™ users then can exchange mail with UNIX users through the VM TCP/IP SMTP gateway. An example of such an environment follows:

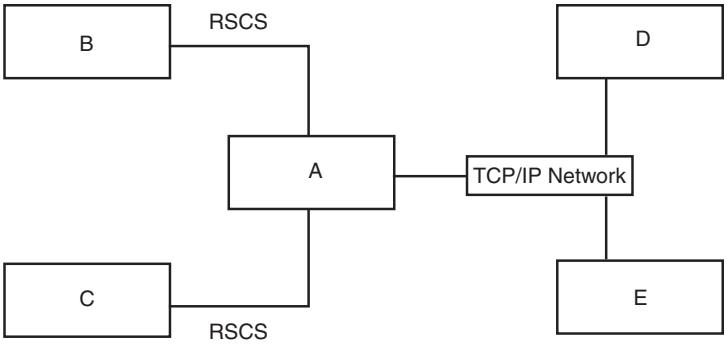


Figure 10. Example SMTP Mail Gateway Environment

Where:

A Is the local VM host, running both TCP/IP and RSCS.

B and C
Are hosts attached to host A through an RSCS network.

D and E
Are hosts attached to host A through a TCP/IP network.

Users on hosts A, B, and C can send mail or files to users on TCP/IP hosts D and E using the CMS NOTE and SENDFILE commands, or PROFS interface. The users on hosts D and E can send mail to the users on host A using addresses in the following format:

© Copyright IBM Corp. 1987, 2016

user_id@A.domain

Where:
user_id Is the user ID of the VM user on host A.
A.domain
Is the TCP/IP host name of host A.

The users on TCP/IP hosts D and E can send mail to users on RSCS hosts B and C using addresses in the following format:

user_id%RSCSHost@A.domain

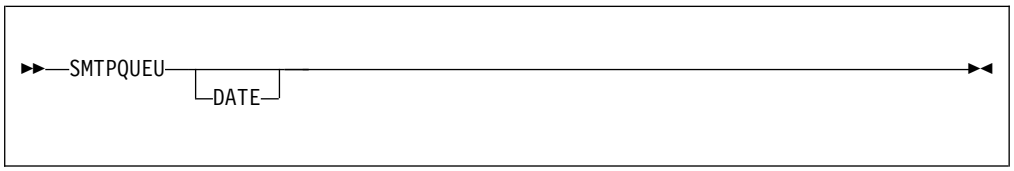
Where:
user_id Is the user ID of the user on the host.
RSCSHost
Is the name of the RSCS host (B or C).
A.domain
Is the TCP/IP host name of host A.

Note and File Delivery

All mail arriving from senders on foreign hosts is delivered to the virtual machine of the VM recipient by the SMTP virtual machine. Normal VM processing places the mail in the virtual reader of the addressee. This mail can be read using OfficeVision or traditional CMS functions such as RDRLIST and PEEK. For information on manipulating electronic mail using CMS, see *z/VM: CMS Commands and Utilities Reference*.

The following command, issued from the CMS command line, checks any mail that SMTP is delivering or waiting to deliver.

SMTPQUEU Command



Purpose

Use the SMTPQUEU command to instruct the SMTP server to produce and return a batch SMTP response file (BSMTP REPLY) that lists the mail queued for delivery to one or more sites. The response file is spooled to the user that issued the SMTPQUEU command.

Operands

DATE
Specifies that the date and time that a mail item was added to a given site delivery queue should be included in the SMTP server response file.

Usage Notes

1. SMTPQUEU requests are directed to the SMTP server defined by the first SMTPSERVERID statement in the TCPIP DATA file. If no such statement exists, requests are directed to the user ID SMTP.

Examples

This example shows the response of the SMTPQUEU EXEC:

```
220-GDLGCT2.ENDICOTT.IBM.COM running IBM VM SMTP Level 540
on Fri, 23 May 2008 1
220 5:10:42 EDT
050 VERB ON
250 Verbose Mode On
050 QUEU DATE
250-Queues on GDLGCT2.ENDICOTT.IBM.COM at 15:57:01 EDT on 05/23/08
250-Spool Queue: Empty
250-Queue for Site: 64.233.185.114 RETRY QUEUE Last Tried: 15:55:24
250-Note 00000002 to <user_123@gmail.com> at 05/23/08 15:08:38
250-Undeliverable Queue: Empty
250-Resolution Queues:
250-Resolver Process Queue: Empty
250-Resolver Send Queue: Empty
250-Resolver Wait Queue: Empty
250-Resolver Retry Queue:
250-00000004 <user_456@hostx.testdomain.com>
250-Resolver Completed Queue: Empty
250-Resolver Error Pending Queue: Empty
250 OK
```

Undelivered Notes

When the SMTP virtual machine cannot deliver a piece of mail, a nondelivery note or an unknown recipient note is forwarded to the sender of the mail explaining the reason for nondelivery. Nondelivery can occur for several reasons. For example, a destination host may not be known, or the recipient may not have a user ID on the destination host.

Nondelivery Note

If a piece of mail cannot be delivered, the body of the original piece of mail is returned as part of the nondelivery notification.

Figure 11 is an example of a nondelivery note.

```
Date: Fri, 8 Jan 93 08:23:54 EST
From: SMTP@VM1.ACME.COM
To:   DANIEL@VM1
Subject: Undeliverable Mail

VM1.ACME.COM unable to deliver following mail to recipient(s):
      <MATT@SMTP-GATEWAY.IBM.COM>
VM1.ACME.COM unable to connect for 3 days to host:
      SMTP-GATEWAY.IBM.COM

       ** Text of Mail follows **
Date: Tue, 5 Jan 93 08:22:36 EST
From: <DANIEL@VM1.ACME.COM>
To:   <MATT@SMTP-GATEWAY.IBM.COM>
Subject:   ACME iron birdseed

Matt,

The shipment of ACME iron birdseed was shipped last Thursday.  Please
advise me if you have not received it, and I'll try to track it down.
Also, your ACME giant rock catapult is on back order; another customer
bought the last one yesterday.

Daniel
```

Figure 11. Example of a Nondelivery Note

Unknown Recipient Note

If a recipient is unknown at the destination host, the destination host does not accept the mail for delivery, and a nondelivery notification is forwarded to the sender.

Figure 12 is an example of an unknown recipient note.

```
Date: Mon, 15 Feb 93 13:32:12 EST
From: SMTP@VM1.ACME.COM
To:   DANIEL@VM1
Subject: Undeliverable Mail

VM1.ACME.COM unable to deliver following mail to recipient(s):
      <GEORGE@SMTP-GATEWAY.IBM.COM>
VM1.ACME.COM received negative reply from host:
      SMTP-GATEWAY.IBM.COM

       ** Text of Mail follows **
Date: Tue, 16 Feb 93 08:22:36 EST
From: <DANIEL@VM1.ACME.COM>
To:   <GEORGE@SMTP-GATEWAY.IBM.COM>
Subject:   Your retirement

George,

I recently learned that you will soon be retiring.  I just wanted to
wish you best of luck and thanks for all the great work you have done.
You were a great asset to your company, and I'm sure they will miss you.

Daniel
```

Figure 12. Example of an Unknown Recipient Note

Using OfficeVision Without OfficeVision Extended Mail Installed

PROFS/OfficeVision is an electronic mail interface for VM users. OfficeVision can send and receive electronic mail from users on TCP/IP networks with OfficeVision Extended Mail. For more information about using OfficeVision Extended Mail, see *PROFS Extended Mail User's Guide and Installation Manual*.

Contact your systems administrator to see if OfficeVision Extended Mail is installed at your site.

Note: Installation of OfficeVision Extended Mail is recommended.

A limited OfficeVision to SMTP interface is provided for sites that do not have PROFS Extended Mail installed. OfficeVision users can prepare mail for TCP/IP network recipients by including the SMTP virtual machine as one of the recipients of the mail. TCP/IP network recipients are specified through a special command within the PROFS note.

Specifying TCP/IP Recipients

When using OfficeVision, you must specify the addresses of RSCS recipients in the following format:

hostname(*user_id*)

Where:

hostname
: Is the host name of the recipient.

user_id
: Is the user ID of the recipient.

This format conforms to the specifications of local and RSCS network addresses, when you are using OfficeVision directly.

You enter the TCP/IP network address using a .ddn command. The following list describes how to use the .ddn command.

- Specify TCP/IP network addresses using the .ddn command, as shown in the following examples.

  Note: The host and user ID pairs may be specified in either one of the following two formats.

  ```
  .ddn TcpHost(TcpUserid)
  .ddn TcpUserid@TcpHost
  ```

- Host names and user IDs can have more than 8 characters. The case (upper or lower) of the TCP/IP network addresses is preserved.
- Code .ddn commands starting in column 1, like all other OfficeVision dot commands.
- Specify multiple addresses with a single .ddn command. At least one blank space must separate each pair of addresses, as shown in the following example.

  ```
  .ddn TcpHost1(TcpUserid1) TcpUserid2@TcpHost2
  ```

- Code as many .ddn statements as necessary. The following example shows a group of three .ddn statements.

  ```
  .ddn TcpHost1(TcpUserid1) TcpHost2(TcpUserid2)
  .ddn TcpUserid3@TcpHost3 TcpUserid4@TcpHost4
  .ddn TcpHost5(TcpUserid5) TcpUserid6@TcpHost6
  ```

- Do not place text from your note on the same line with a .ddn command in column one. This text would be mistakenly interpreted as additional addresses.
- No embedded blanks can appear within a TCP/IP address.
- PROFS does not convert addresses specified on the .ddn command line to uppercase.
- .ddn commands are not removed from the text of the note, when the note is sent to other PROFS users on the local or RSCS attached systems. The .ddn commands are removed from copies sent to the TCP/IP network recipients.
- If you resend a note with .ddn commands to SMTP, all of the recipients specified on the .ddn command receive a copy of the mail.
- SMTP ignores .ddn commands in a forwarded note.

Example of Sending a Note Copy to SMTP

OfficeVision does not deliver the mail to the TCP/IP network recipient; you must send a copy of the PROFS note to the SMTP virtual machine. SMTP reads the addresses of the recipients from the OfficeVision note and delivers the note to each recipient. For example,

```
 E34                                Send A Note
 Send to: smtp user2 system3(user1)
 From: MyName
 Subject: (U)
00001 .ddn yktvmv.watson.ibm.com(user20) user10@ytkvmv.watson.ibm.com
00002
00003 ...Text of note goes here...
00004
00005
00006
```

To verify that SMTP receives a copy of the mail that you send, specify the address of the SMTP virtual machine in the *Send To:* field of the note, or with a .cc or .ad command.

The address of the SMTP virtual machine is SMTP, unless your system administrator tells you otherwise.

Note Delivery

When SMTP receives a note from a OfficeVision user, the note is rewritten with a SMTP mail header. The note is also placed in batch SMTP format.

If any of the TCP/IP addresses specified are invalid, SMTP sends an error message to the PROFS user. This error message includes the batch SMTP version of the PROFS note. If the TCP/IP addresses are valid, but the mail cannot be delivered, the PROFS user receives an error message of the type described in "Undelivered Notes" on page 121.

Using IMAP to Access Electronic Mail

If you are authorized to access mail from an installation that is utilizing the z/VM IMAP server support, you as a mail user can use an IMAP client to access your mail that is residing in the IMAP mailstore on z/VM. This permits you, (using a "client" email program) to access your mail as if it were local. For example, email stored on the z/VM IMAP server can be manipulated from a desktop computer at home, a workstation at the office, or a laptop computer while away on business without the need to transfer messages back and forth.

In order to access mail from an IMAP server, the following needs to take place:

- The user (client) must be enrolled into the IMAP Mailstore.
- The client email software must be configured to recognize the IMAP server in order to access the message folders.
- The client email software must connect to the z/VM IMAP server over an IPv4 connection.

You will need to contact your TCP/IP administrator in order to be enrolled as an IMAP user and can read further about IMAP in the *z/VM: TCP/IP Planning and Customization*. You will also need to refer to your particular email client's documentation on how to access your mail folders.

Using Secure SMTP

The SMTP virtual machine may be configured by your TCP/IP administrator for Secure SMTP. This would mean that when sending and receiving mail across a TCP/IP network, SMTP will attempt to use secure connections. The method used for securing SMTP connections is known as Transport Layer Security (TLS) and is documented in RFC 3207 (SMTP Extension for Transport Layer Security). When the server is configured to support TLS, a negotiation will take place between the SMTP client and the SMTP server to secure the connection, and all data that is sent across that connection is encrypted using Secure Socket Layer (SSL). From an end user's perspective, the fact that the mail is flowing across secure connections is transparent.

Using the SMSG Interface to SMTP

The VM Special Message Facility (SMSG) command provides an interactive interface to the SMTP virtual machine to perform general user tasks, such as:

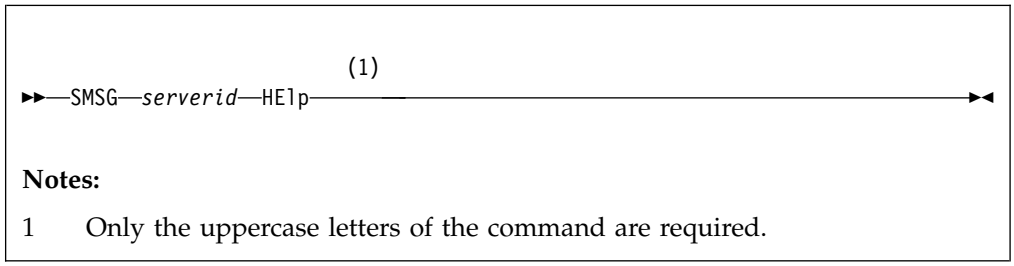
- Querying the SMTP mail delivery queues as well as the operating statistics of the SMTP virtual machine
- Performing privileged system administration tasks, such as rebooting or shutting down the SMTP virtual machine and enabling or disabling various tracing and debugging options.

Note: There is a privileged user SMSG command set designed for system administration tasks which are located in *z/VM: TCP/IP Planning and Customization*. Responses to commands are sent back to the originator of the command using CP MSG commands (or CP MSGNOH commands if SMTP is running with CP privilege class B).

General User SMSG Commands

The following are SMSG commands that the general user can use.

SMSG HELP Command



Notes:

1 Only the uppercase letters of the command are required.

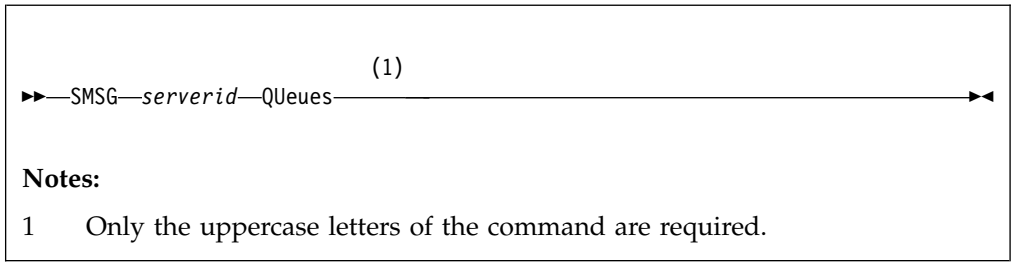
Purpose

Use the SMSG HELP command to provide a list of valid SMSG commands accepted by SMTP.

Operands

serverid
Specifies the user ID of the virtual machine running the z/VM SMTP server.

SMSG QUEUES Command



Notes:

1 Only the uppercase letters of the command are required.

Purpose

Use the SMSG QUEUES command to provide a list of mail queued on the various SMTP mail delivery queues.

Operands

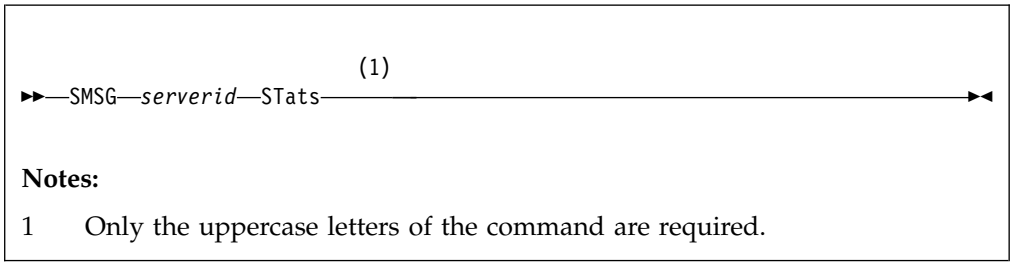
serverid
Specifies the user ID of the virtual machine running the z/VM SMTP server.

Example

To get a list of queued mail on the SMTP mail delivery queues, enter:

```
 smsg smtp qu
Ready; T=0.01/0.01 10:46:13
 * From SMTP: ----- Mail Queues -----
 * From SMTP: Spool Queue:         0
 * From SMTP: Undeliverable Queue: 0
 * From SMTP: --- Resolver Queues ---
 * From SMTP: Process Queue:       0
 * From SMTP: Send Queue:          0
 * From SMTP: Wait Queue:          1
 * From SMTP: Retry Queue:         1
 * From SMTP: Completed Queue:     0
 * From SMTP: Error Queue:         0
```

SMSG STATS Command



Notes:

1 Only the uppercase letters of the command are required.

Purpose

Use the SMSG STATS command to provide operating statistics about the SMTP virtual machine. These statistics include:

- The date that SMTP was last booted
- The number of spool files in SMTP's RDR
- The number of spool files in SMTP's RDR in HOLD status (these spool files are too big for SMTP to process)
- The number of files on SMTP's 191 minidisk
- The percent of disk space in use on SMTP's 191 minidisk
- The statistics about mail handled by SMTP over the past two days. These statistics include:
  - The number of pieces of mail that arrived over tcp connections
  - The number of pieces of mail that arrived from spool (from local or RSCS senders)
  - The number of pieces of mail generated in response to requests to VERBose batch SMTP connections
  - The number of pieces of mail generated to return error mail to the sender

- The number of pieces of mail delivered to local recipients
- The number of pieces of mail delivered to recipients on the RSCS network
- The number of pieces of mail delivered to recipients on the TCP/IP network
- The number of TCP connections opened through which mail was received
- The number of TCP connections opened through which mail was delivered

Operands

serverid
Specifies the user ID of the virtual machine running the VM SMTP server.

Example

To obtain statistic information about the SMTP virtual machine, enter:

```
 smsg smtp stat
Ready; T=0.01/0.01 10:46:37
 * From SMTP: Last Up Time: Mon, 13 Dec 93 09:11:30 EST
 * From SMTP: Spool Files :    0  Held:    0
 * From SMTP: Disk Files  :    6  Full:   01%
 * From SMTP: Statistics  : 12/23  12/22  12/21  12/20
 * From SMTP: From TCP    :    23     45     44     50
 * From SMTP: From Spool  :     1      3      7     36
 * From SMTP: BSMTP Logs  :    10     22     22     24
 * From SMTP: Error Mail  :     2      4      3      8
 * From SMTP: To Local    :    65    153    148    166
 * From SMTP: To RSCS     :     0      0      0      4
 * From SMTP: To TCP      :     0      1      6     30
 * From SMTP: Passive Opns:    22     43     45     50
 * From SMTP: Active Opens:     0      1      5     13
```

Receiving Electronic Mail on VM

When a CMS user receives electronic mail transmitted over a TCP/IP network, it is held for the user in their virtual reader until the user acts on it. If the TCPIP DATA file has been set up with SMTPSERVERID definitions for the SMTP server(s), CMS productivity aids that manipulate reader files will recognize the mail's true origin as other than the local SMTP userid. RECEIVE and PEEK will issue messages with the domain name origin of the mail. For RECEIVE, this message is limited to 240 characters and will be truncated if the domain name causes the message text to exceed that limit. For PEEK, this message is limited to the user's defined screen width and will be truncated if the domain name causes the message text to exceed that limit. For electronic mail received with the LOG option, the user's NETLOG file will also contain the domain name of the originator.

The origin of TCP/IP-based e-mail is also identified in the User and Node columns of the RDRLIST panel. Because these fields are each limited to eight characters, this origin information is divided into user and host domain information. Still, truncation of these values is likely, and is performed as described in the next section.

The most widely-used form of an origin address is:

```
user@host.domain
```

The RDRLIST “Node” value is obtained from the *host.domain* portion of the origin address. This information is tokenized using the periods (.) in this information as delimiters, and the displayed “Node” value is the most complete string of

unaltered tokens that can be represented using eight characters. If *host* is itself greater than eight characters, its left-most eight characters of are displayed.

The RDRLIST "User" value is obtained from the *user* portion of the origin address that precedes the "@" sign. If *user* is greater than eight characters long, its left-most eight characters are displayed in the "User" field. However, if *user* contains any periods (.), this value is instead tokenized and then displayed in the same fashion as is the *host.domain* information.

If mail is sent using source routing, the origin address contains additional information, and would be similar to:

```
otherhost.other.domain.:user@host.domain
```

If such an address is encountered, data to the left of the last colon (:) present is discarded, and the RDRLIST "User" and "Node" values are obtained from the remaining origin address information, as previously described.

It is also possible for the origin address to include a percent (%) sign, and be of the form:

```
user%host@domain
```

If an address of this kind is encountered, the "@" sign is effectively replaced with a period (.), and the resulting *user* and *host.domain* values are processed to obtain the RDRLIST "User" and "Node" values in the usual way.

Here are some examples of domain name addresses and the resulting "User" and "Node" information that would be displayed on the RDRLIST panel:

```
john.doe%system1@vnet.ibm.com
  becomes  User: john.doe
  and      Node: system1

joneswk@gny1vm.vnet.ibm.com
  becomes  User: joneswk
  and      Node: gny1vm

Tom_Duffington@arbitrary_host.isp.com
  becomes  User: Tom_Duff
  and      Node: arbitrar

music.marist.edu:urmm@vm.marist.edu
  becomes  User: urmm
  and      Node: vm

pink.floyd@darkside.moon.com
  becomes  User: pink
  and      Node: darkside

p.t.barnum@big.top.circus.com
  becomes  User: p.t
  and      Node: big.top
```

Chapter 5. Logging On to a Foreign Host

The Telnet and TN3270 protocols provide a standardized interface that allows fullscreen and linemode terminal oriented processes on hosts that support TCP/IP to communicate with each other.

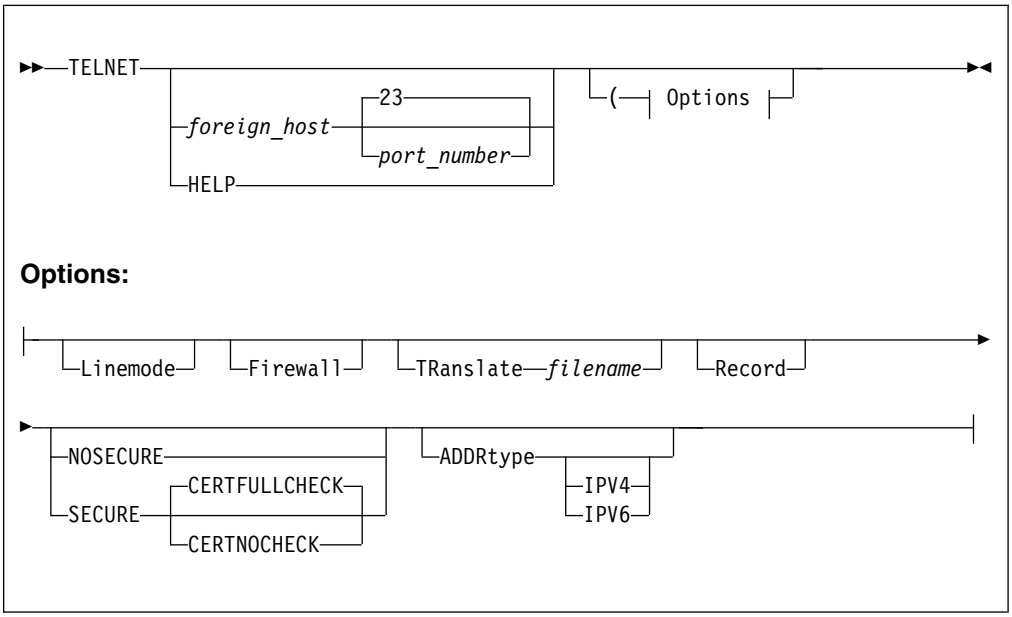
This chapter describes the following subjects:

- Using the TELNET Command
- Using the TELNET Function Keys
- Using the TELNET Subcommands
- Sending ASCII Control Characters to a Host in Line Mode.

TELNET Command

Purpose

Use the TELNET command to log on to a foreign host supporting TCP/IP.



Operands

foreign_host
Specifies the name or internet address of the foreign host. The internet address may be an IPv4 or an IPv6 address.

If you do not specify the name or internet address of the foreign host, you are prompted for the *foreign_host*.

Note: Attempting to specify an IPv6 link local address will produce unpredictable results.

port_number
Specifies the port number to which you want to connect on the foreign host.

© Copyright IBM Corp. 1987, 2016

When connecting to a non-TELNET port, the data exchange must follow the protocol recognized by the other port. The default is well-known port 23.

Linemode
Indicates the logon operation. LINEMODE uses the line mode and prevents operation in the transparent (TN3270) mode.

In line mode, the foreign host's output is displayed on your screen one line at a time, without full-screen capabilities. In transparent (TN3270) mode, the foreign host's full-screen capabilities are functional on your local terminal.

Firewall
Specifies that a connection is to be made to a host in transparent (TN3270) mode through a proxy server. TELNET establishes a line-mode connection to the designated foreign host (the proxy server). You conduct a conversation with the proxy server to cause it to establish a connection to the actual destination host. TELNET negotiates the termnial type with that host and will enter 3270 mode if permitted.

TRanslate *filename*
Specifies a nonstandard translation table file. If this parameter is not specified, TELNET searches sequentially for TELNET TCPXLBIN and STANDARD TCPXLBIN. If neither is found, TELNET uses the compiled translation table. TELNET TCPXLBIN is supplied. The file type is always TCPXLBIN.

A nonstandard translation table is used in line mode only.

Record
Creates a log of interactions with linemode hosts in FILE LOGFILE A. A different file can be selected by issuing a CMS FILEDEF for ddname LOGFILE.

SECURE
A secure telnet connection will be attempted if the telnet server supports secure connections. If the telnet does not support secure connections or if an SSL server is not running on the local system, the telnet command will fail. Specifying the SECURE option overrides a SECURETELNETCLIENT NO statement specified on the TCPIP DATA file.

CERTFULLCHECK
The remote server certificate and any associated CA certificates are verified for correctness for the connection. If a certificate error is detected, the corresponding connection fails.

A self-signed certificate cannot be used for secure connections unless it has been added to the local certificate database. For more information, see Using Self-Signed Certificates in *z/VM: TCP/IP Planning and Customization*.

CERTNOCHECK
The CERTNOCHECK option is accepted and ignored. All connections are handled as described for the CERTFULLCHECK operand.

NOSECURE
A clear telnet connection will be attempted. If the telnet server does not support clear connections, the telnet command fails. This is the default taken when neither SECURE or NOSECURE is specified, unless overriden by SECURETELNETCLIENT YES in the TCPIP DATA file. Specifying the NOSECURE option overrides the SECURETELNETCLIENT YES statement specified in the TCPIP DATA file.

ADDRTYPE IPV4
ADDRTYPE IPV6
Specifies which type of internet address Telnet will attempt to resolve a host

name to. If ADDRTYPE is not specified, Telnet will default to IPv4 for secure connections. For connections which are not secure, Telnet will use the first address returned by the resolver whether it is IPv4 or IPv6.

Usage Notes

- The TELNET command can only be used when logged on to VM using a 3270-type display device. It cannot be used from a linemode terminal.
- When you use the TELNET command to connect to a foreign host running TCP/IP, your foreign terminal session resembles a local terminal session.
- When Telnetting to a 2074 controller the Firewall option must be specified. Also, the TN3270E LU name and printer functions cannot be used.

TELNET Function Keys

This section describes the functions that are assigned to Program Function (PF) keys when you invoke TELNET in transparent mode and line mode.

In Transparent (TN3270) Mode

In transparent (TN3270) mode, the only function key that is available is the PA1 attention key. The PA1 key in transparent mode indicates that you want to invoke a TELNET subcommand.

For information about how to send the PA1 keystroke to the foreign session, see "PA1" on page 134.

In Line Mode

Table 19 shows the functions that are assigned to PF keys when you invoke TELNET in line mode.

Table 19. TELNET PF Key Functions

Key	Function
PF4-PF12, PF16-PF24,	Displays the TELNET subcommand prompt.
PF1, PF13	Retrieves the previous input line.
PF2, PF14	Scrolls the screen forward halfway.
PF3, PF15	Turns off the display of the user information line. Use either of these keys before entering your password.

Note: When you connect to a VM host from a non-VM host, the following applies:

- If you invoke TELNET from a non-VM host in order to connect to a VM host and your client cannot emulate a 3270 data stream, transparent mode will not be possible. Instead, you will be connected to VM as a line-mode, start-stop terminal.
- When connected to VM as a line-mode, start-stop terminal, two TELNET subcommands have a special meaning:
  - The Abort Output (AO) subcommand causes a single attention to be presented. A single attention displays a VM READ.
  - The Interrupting Process (IP) subcommand causes a double attention to be presented. A double attention displays a CP READ.

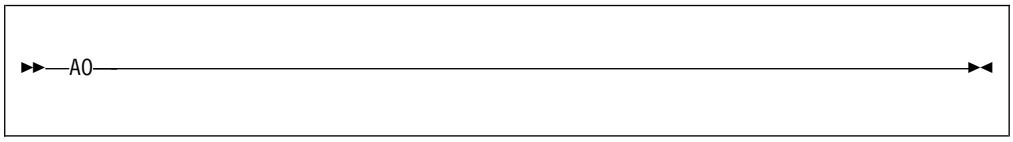
TELNET Subcommands

The following TELNET subcommands are described in detail.

To invoke a TELNET subcommand while you are logged on to the foreign host, press the designated PF key. For a list of the designated PF keys, see Table 19 on page 131. After you press the PF key, you are prompted to enter the TELNET subcommand. TELNET subcommands can be entered in uppercase or lowercase.

You can use the PA1, AYT, AO, IP, and SYNCH subcommands to communicate with the foreign host while you are logged on with the TELNET command. The following sections describe these subcommands.

AO



Purpose

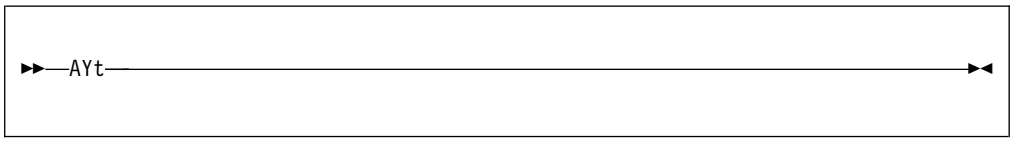
Use the AO (Abort Output) subcommand to stop displaying output.

The AO subcommand is useful to clear any output that has already been produced but has not been displayed on your terminal.

Operands

The AO subcommand has no parameters.

AYT



Purpose

Use the AYT (Are You There) subcommand to query the existence of the connection.

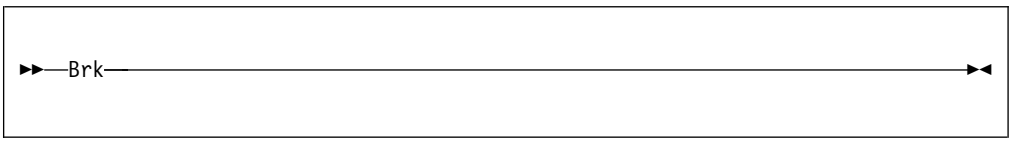
Operands

The AYT subcommand has no parameters.

Usage Notes

1. If the connection exists and you are operating in transparent mode, the terminal makes a sound. If you are operating in line mode, you receive a message from the Telnet server.

BRK



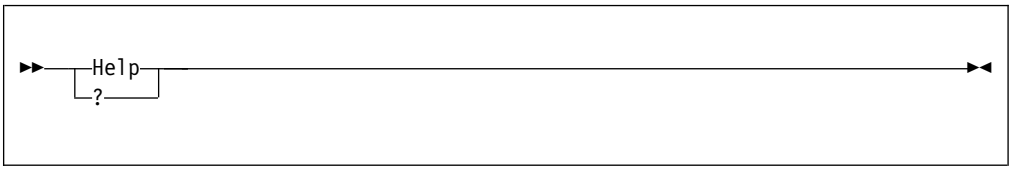
Purpose

Use the BRK subcommand to send a Break or Attention (Attn) keystroke to the remote session.

Operands

The BRK subcommand has no parameters.

HELP



Purpose

Use the HELP or ? subcommand to get help.

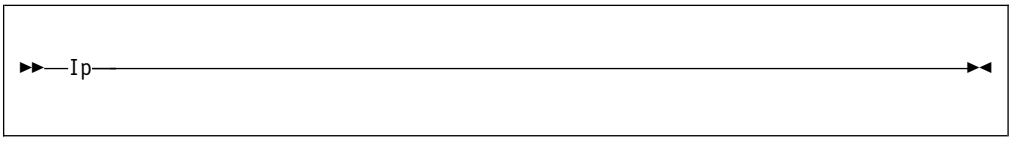
Operands

The HELP subcommand has no parameters.

Usage Notes

1. When you invoke the HELP or ? subcommand, a list of TELNET subcommands is displayed.

IP



Purpose

Use the IP (Interrupting Process) subcommand to interrupt the current process running on the foreign host.

Operands

The IP subcommand has no parameters.

Usage Notes

1. The IP subcommand is useful when you want to stop a process that is in a loop, or when you want to stop a process that you inadvertently started.

PA1

Purpose

Use the PA1 subcommand to send a PA1 keystroke to the remote session in transparent mode.

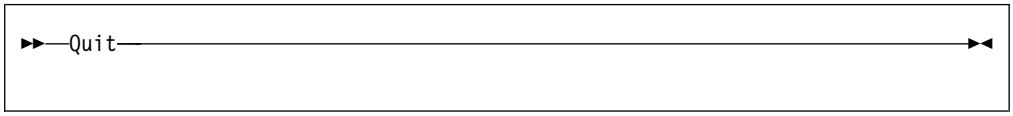
Operands

The PA1 subcommand has no parameters.

Usage Notes

1. The PA1 subcommand operates only in transparent mode. The PA1 subcommand replaces the PA1 attention key on the foreign host.

QUIT



Purpose

Use the QUIT subcommand to end the TELNET session.

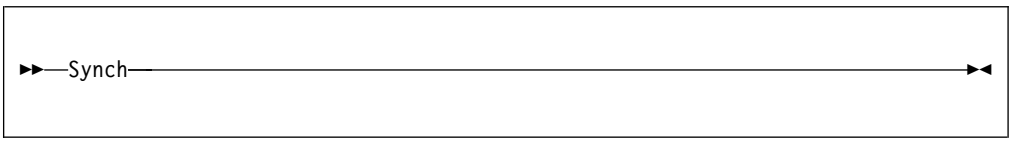
Operands

The QUIT subcommand has no parameters.

Usage Notes

1. If you are connected to a foreign host running TCP/IP and you use the QUIT subcommand, you are disconnected but not logged off the foreign host. The Virtual Telecommunication Access Method (VTAM®) application you are accessing determines whether you are also logged off from the foreign host. For example, the Time Sharing Option (TSO) application disconnects you, but does not log you off.
2. When you want to end a logon session with the foreign host, use the logoff procedure for the host.

SYNCH



Purpose

Use the SYNCH (Synchronize data path) subcommand to clear the data path.

Operands

The SYNCH subcommand has no parameters.

Usage Notes

1. The SYNCH subcommand clears the data path to the foreign host, except for any TELNET subcommands in the data path.

Sending ASCII Control Characters to a Host in Line Mode

In line mode, use the cent sign (¢) or the grave accent (`) to indicate a control character. For example, to send Ctrl-p, use either ¢p or `p.

Other control characters are shown in Table 20.

Table 20. Control Characters

Characters	ASCII Output
`2 - `6	1B-1F
`#	FF (DEL)
`{	5B (left square bracket)
`}	5D (right square bracket)

You use either ¢ or ` before pressing the Enter key to suppress the sending of a carriage return and line feed. This function is useful for continuing a line without introducing a new line.

The following is an example of using the ¢ to continue a line.

```
PURGE RDR 45 78 99 67 69¢Enter
56 44
```

This function works if the command environment of the foreign host responds to a single non-Enter character, without a carriage return and line feed following it.

Chapter 6. Using the LDAP Client Utilities

The LDAP utilities provide a way to add, compare, modify, search, and delete entries in any server accepting LDAP protocol requests.

Each of the following utilities can be run from CMS:

ldapchangepwd
ldapcompare
ldapdelete
ldapadd
ldapmodify
ldapmodrdn
ldapsearch

Each utility accepts many possible parameters. See "Using the Command Line Utilities" on page 138 for a complete explanation of the parameters that can be supplied to each of the client utility programs.

Running the LDAP Client Utilities in CMS

The LDAP client utilities can be run from CMS. In order to do this, you need access to the TCPMAINT 592 disk where the LDAP utility EXECs, load modules, and message catalogs reside.

By default, the LDAP client utilities will access the US English message catalogs. If you want the utilities to access the Japanese message catalogs instead, set the NLSPATH environment variable by executing the following commands from a REXX EXEC:

```
nlspath = '/usr/lib/nls/msg/Ja_JP.IBM-939/%N'
'globalv select cenv put nlspath'
```

Running these utilities follows the same syntax as would be used if running them in other systems, except that the program names are invoked using EXECs, so the invocation names are eight characters or less. To run these utilities from CMS, use the following names:

Executable	CMS name
ldapchangepwd	LDAPCHPW
ldapadd	LDAPADD
ldapcompare	LDAPCMPR
ldapdelete	LDAPDLET
ldapmodify	LDAPMDFY
ldapmodrdn	LDAPMRDN
ldapsearch	LDAPSRCH

© Copyright IBM Corp. 1987, 2016

Throughout this information, whenever an executable is mentioned, it implies that you use the CMS EXEC name to invoke it. For example, whenever the information discusses **ldapcompare**, you should invoke the executable with LDAPCMPR.

If you are using the utilities in interactive mode (for example, reading DNs, `changetype` lines, and so on, from standard input), you can break out of interactive mode by typing HX and pressing the Enter key. Doing this returns you to CMS. This is similar to pressing Ctrl+C keys in UNIX.

Using the Command Line Utilities

The **ldapchangepwd**, **ldapcompare**, **ldapdelete**, **ldapmodify**, **ldapadd**, **ldapmodrdn**, and **ldapsearch** utilities all perform a bind operation to the LDAP server. When bind is invoked, several results can be returned. Following are bind results using various combinations of user IDs and passwords.

1. If a null or zero length DN is specified, the user receives unauthenticated access.
2. If a non-null, non-zero length DN is specified, a password must also be specified.
   - If the DN falls outside the scope of the suffixes managed by the server, the DN must match one of the **adminDN**, **masterServerDN**, or **peerServerDN** configuration options specified in the server configuration file, and the password must match the corresponding **adminPW**, **masterServerPW**, or **peerServerPW** configuration option. In this case, the user is bound as the LDAP server root administrator or as the master or peer replica administrator.
   - If the DN falls within the scope of a suffix managed by the server, then there must be an entry in the server directory for that DN. The password specified by the user must match a password associated with the entry. The user is then bound with that identity. If the DN also matches one of the **adminDN**, **masterServerDN**, or **peerServerDN** configuration options specified in the server configuration file, then the user is bound as the LDAP server root administrator or as the master or peer replica administrator. If the DN has been assigned one or more administrative roles, then the user is bound with those administrative roles. See "Administrative group and roles" in *z/VM: TCP/IP LDAP Administration Guide* for more information about administrative roles.

An error is returned when binding with any other combination of user ID and password.

Note: If you are using an LDAP server other than the z/OS or z/VM LDAP server, the bind results might be different.

Use of BFS Files

If a BFS file is specified for the -f option, or the -K option is used to specify an SSL key database file, the LDAP utilities will make use of OpenExtensions services to access those files. This may have implications if these utilities are called from an application program. See "Running OpenExtensions Applications" in *z/VM: OpenExtensions User's Guide* for more information.

Using Special Characters with the LDAP Utilities

LDAP utilities often require special characters such as " (double quotation mark) or @ (at). Unless your system administrator has changed them, these characters are also used by CP to perform command line editing. To see which line editing characters you are using, issue the CP QUERY TERMINAL command. If an LDAP utility filter or other operand uses any of the special characters shown in the output of the query, you need to do one of the following:

- Turn off line editing by issuing the CP SET LINEDIT OFF command.
- Disable CP's scan for the special character by issuing the CP TERMINAL *xxxxx* OFF command, where *xxxxx* is LINEND, LINEDEL, CHARDEL, or ESCAPE, as needed.
- Use the ESCAPE character as a signal to CP that the character *immediately following* the escape character is to be used as-is. To determine the ESCAPE character you are using, issue the CP QUERY TERMINAL command.

Example: Your escape character is set to " (double quotation mark) and you want to issue the command:

```
ldapsrch -h ldap.company.com -b "ou=ldap,o=company.com" "cn=john doe"
```

You should issue CP TERMINAL ESCAPE OFF prior to issuing the command. If you do not do so, CP removes the double quotation marks before the command executes, resulting in errors. Alternatively, you could issue CP SET LINEDIT OFF.

Specifying a value for a filename

When running the **ldapcompare**, **ldapdelete**, **ldapmodify**, **ldapadd**, **ldapmodrdn**, and **ldapsearch** utilities, the file option (-f) value can be specified as follows:

/pathname/filename
: Specifies the full path name of a file in the BFS file system.

filename
: Specifies a path name that is relative to the current working directory of the LDAP client utility.

//filename.filetype.filemode
: Specifies a CMS file. **filemode** is optional; the default is A.

SSL/TLS information for LDAP utilities

The contents of a client's key database file is managed with GSKKYMAN (the **gskkyman** utility). For information about GSKKYMAN, see Chapter 8, “SSL Certificate/Key Management and SSL Tracing Information,” on page 233. GSKKYMAN is used to define the set of certification authorities (CAs) that are to be trusted by the client. By obtaining certificates from trusted CAs, storing them in the key database file, and marking them as trusted, you can establish a trust relationship with LDAP servers that use certificates issued by one of the CAs that are marked as trusted.

If the LDAP servers accessed by the client use server authentication, it is sufficient to define one or more trusted root certificates in the key database file. With server authentication, the client can be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL/TLS connection with the server are encrypted, including the LDAP credentials that are supplied on the bind.

For example, if the LDAP server is using a high-assurance VeriSign certificate, you should obtain a CA certificate from VeriSign, receive it into your key database file, and mark it as trusted. If the LDAP server is using a self-signed GSKKYMAN server certificate, the administrator of the LDAP server can supply you with a copy of the server's certificate request file. Receive the certificate request file into your key database file and mark it as trusted.

Using the LDAP client utilities without the **-Z** parameter and calling the secure port on an LDAP server (in other words, a non-secure call to a secure port) is not supported. Also, a secure call to a non-secure port is not supported.

SSL/TLS encrypts the key database file. Either the password must be specified as part of the **-P** parameter or a file specification of a stash file that was created using GSKKYMAN must be specified (the form is **file://** followed immediately—no blanks in between—by the file specification of the stash file).

The file names specified on the **-K** (SSL key database file) and **-P** (SSL password stash file) parameters must refer to BFS files. This requirement means that if you want to use a secure connection to the LDAP server, you must have access to a BFS in which to store the SSL key database file, or the SSL password stash file, or both.

Note: If an error occurs when using the LDAP client utilities with the **-Z** parameter, the utility will return a failureReasonCode, which is the SSL reason code. These reason codes are documented in the LDAPSSL H file found on the TCPMAINT 592 disk.

SSL initialization failure

If SSL initialization fails, an error message similar to the following is returned:

```
ldap_ssl_client_init failed! rc == 113, failureReasonCode == 2
reason text: SSL initialization failed
```

The failureReasonCode indicates the cause of the SSL failure and is mapped from the return code of various SSL functions. See Table 21 for these values.

Table 21. SSL failure reason codes

Failure reason code	SSL return code	Failure reason code description
-1	402	No ciphers matched the server and client lists of acceptable ciphers
-2	403	No client certificate is to be used
-6	405	The certificate type is not supported
-10	406	I/O error communicating with peer application
-11	410	Incorrectly-formatted message received from peer application
-12	411	Message verification failed
-13	412	SSL protocol or certificate type is not supported
-14	413	Certificate signature is not correct for a certificate received from the peer
-15	414	Certificate is not valid
-16	415	Peer application has violated the SSL protocol
-17	416	Not authorized to access key database

Table 21. SSL failure reason codes (continued)

Failure reason code	SSL return code	Failure reason code description
-18	417	Self-signed certificate cannot be validated
-20	4	Insufficient storage is available
-21	5	The environment or connection is not in the open state
-22	420	Socket closed by peer
-41	422	V3 cipher is not valid
-99	12 or any other unmapped SSL reason code	Unrecognized error
-1000	none	Failed loading SSL DLL
-1001	none	Failed locating SSL function
1	102	Keyring I/O error
2	202	Keyring open error
4	408	Keyring password is incorrect
12	6, 407	Keyfile label is not valid or certificate is not trusted
106	106	Key database file is corrupted
109	109	Key database does not contain any valid CA certificates
201	201	Key database password or stash filename not set
203	203	Unable to generate temporary RSA key
204	204	Key database password is expired
301	301	Close failed
302	302	Connection has an active write
401	401	Validity time period for the certificate has expired
427	427	Unable to access the LDAP directory
428	428	The client key did not contain a private key
431	431	Certificate has been revoked
432	432	Session renegotiation is not allowed
433	433	Key exceeds allowable export size
434	434	Certificate key is not compatible with the negotiated cipher suite
435	435	Missing CA certificate
436	436	CRL cannot be processed
437	437	A close notification alert has been sent for the connection
438	438	Internal error reported by remote partner
439	439	Unknown alert received from remote partner
501	501	The buffer size is negative or zero
502	502	Operation would block
503	503	Read would be blocked
504	504	Write would be blocked

Table 21. SSL failure reason codes (continued)

Failure reason code	SSL return code	Failure reason code description
505	505	Record overflow
602	602	Function identifier is not valid
701	701	Attribute ID is not valid
702	702	Attribute length is not valid
703	703	Attribute enumeration value is not valid
705	705	Attribute value is not valid
706	706	Attribute parameter value is not valid
10001	1	Environment or SSL handle not valid
10003	3	Internal SSL error
10007	7	No certificate received from partner
10008	8	Certificate validation error
10009	9	Error processing cryptography
10010	10	Error validating ASN.1 fields in certificate
10011	11	Error connecting to LDAP server
10103	103	The database is not a key database

Using Environment Variables to Control SSL/TLS Settings

The LDAP client utilities do not support SSL V2 protocol, and disable it from being used. SSL V3, TLS V1.0, TLS V1.1, and TLS V1.2 protocols are supported. The z/OS System SSL defaults and environment variables control which of these supported protocols are enabled or disabled, and which cipher specifications apply. For example, the environment variable `GSK_PROTOCOL_SSLV3` can be set to "ON" to enable SSL V3 protocol, or "OFF" to disable SSL V3 protocol. The environment variable `GSK_PROTOCOL_TLSV1` can be set to "ON" to enable TLS V1.0 protocol, or "OFF" to disable TLS V1.0 protocol.

TLS V1.1 and TLS V1.2 protocols are disabled by default. To enable these protocol levels or to override the default cipher specifications, the z/OS System SSL environment variables can be used.

- Set GSK_PROTOCOL_TLSV1_1 "ON" to enable TLS V1.1 protocol.
- Set GSK_PROTOCOL_TLSV1_2 "ON" to enable TLS V1.2 protocol.
- Choose which cipher format is appropriate. Note that only one set of cipher suite specifications (2-byte or 4-byte) is applicable, depending on the setting of the `LDAP_SSL_CIPHER_FORMAT` environment variable.
  - If the default 2-byte cipher suites are sufficient, you can allow the settings to default. If you want to override the cipher suite specifications, and all can be specified as 2-byte cipher suite values, you can set `GSK_V3_CIPHER_SPECS` to override the default cipher specifications, specifying the set of 2-byte values you want. You may also set the `LDAP_SSL_CIPHER_FORMAT` environment variable to `CHAR2`, or do not set it.
  - If you require any cipher suites that can only be specified as a 4-byte value, you must set `GSK_V3_CIPHER_SPECS_EXPANDED` to override the default cipher specifications, specifying the set of 4-byte values you want. You must also set the `LDAP_SSL_CIPHER_FORMAT` environment variable to `CHAR4`.

- Set GSK_SUITE_B_PROFILE to the value you want to apply Suite B-compliant options for your SSL connection. See z/OS Cryptographic Services System Secure Sockets Layer Programming for more information. Note that enabling Suite B by using this environment variable causes the settings of the other environment variables noted above to be ignored, which includes the LDAP_SSL_CIPHER_FORMAT.

Enabling Tracing

You can enable tracing by setting the LDAP_DEBUG environment variable. In CMS, environment variables are set with the GLOBALV SELECT command, specifying the CENV group. For information on the setting environment variables, see "Using Environment Variables" in *z/VM: TCP/IP Programmer's Reference*.

The value for LDAP_DEBUG is a mask that you can specify as follows:

- A decimal value (for example, 32).
- A hexadecimal value (for example, x20, X20, 0x20, or 0X20)
- A keyword (for example, FILTER)
- A construct of those values using plus and minus signs to indicate inclusion or exclusion of a value. If the construct starts with a plus or minus sign, the current trace level is modified. Otherwise, the current trace level is set to 0 before processing the trace values.

Restrictions: To enable or change tracing using this method, the LDAP_DEBUG environment variable must be set or changed before the client runtime is first initialized. Later changes to the value of LDAP_DEBUG have no effect on the debug level of the client.

Examples: The following are examples of values for LDAP_DEBUG:

- Specifying 32768+8 is the same as specifying 32776, or x8000+x8, or ERROR+CONNS
- Specifying 2146959359 is the same as specifying ANY-STRBUF

The LDAP trace routine uses the IBM-1047 code page when writing text data to the trace data set. The trace output is written to stdout unless the LDAP_DEBUG_FILENAME environment variable is defined. If the application spawns additional processes, specify % as part of the trace file name. This causes the % to be replaced by the current process identifier, thus generating a unique filename for each process. Failure to do this can cause the trace file to be corrupted because locking is done on a process basis.

Example: The following example shows the use of % in the trace file name.

```
GLOBALV SELECT CENV SET LDAP_DEBUG_FILENAME myapp.%.trc
```

Table 22 lists the debug levels and related decimal, hexadecimal and keyword values. Keywords can be abbreviated using the uppercase characters for each keyword.

Table 22. LDAP trace debug levels

Keyword	Decimal	Hexadecimal	Description
OFF	0	0x00000000	No debugging
TRACe	1	0x00000001	Routine entry and exit
PACKets	2	0x00000002	Packet activity
ARGS	4	0x00000004	Data arguments from requests

Table 22. LDAP trace debug levels (continued)

Keyword	Decimal	Hexadecimal	Description
CONNs	8	0x00000008	Connection activity
BER	16	0x00000010	ASN.1 encoding and decoding
FILTer	32	0x00000020	Search filters
MESSage	64	0x00000040	Message activity and events
ACL	128	0x00000080	Access control list activity
STATs	256	0x00000100	Operational statistics
THREad	512	0x00000200	Threading activity
REPLication	1024	0x00000400	Replication activity
PARSe	2048	0x00000800	Parsing activity
PERFormance	4096	0x00001000	Backend performance statistics
SDBM	8192	0x00002000	RACF backend activity
REFErral	16384	0x00004000	Referral activity
ERROr	32768	0x00008000	Error conditions
SYSPlex	65536	0x00010000	Sysplex/WLM activity
MULTIServer	131072	0x00020000	Multiple server activity
LDAPBE	262144	0x00040000	Frontend-backend connection activity
STRBuf	524288	0x00080000	NLS and UTF-8 activity
TDBM	1048576	0x00100000	Relational backend activity
SCHEma	2097152	0x00200000	Schema activity
BECApabilities	4194304	0x00400000	Backend capabilities
CACHe	8388608	0x00800000	Cache activity
INFO	16777216	0x01000000	Informational messages
LDBM	33554432	0x02000000	File backend activity
ANY	2147483647	0x7FFFFFFF	All debug levels
ALL	2147483647	0x7FFFFFFF	All debug levels

LDAP URLs and LDAP Filters

An LDAP URL has the following format:

[<][URL:]*scheme*://[*host*[:*port*]][/*dn*[?*attributes*[?*scope*[?*filter*]]]][>]

The URL can optionally be enclosed in angle brackets or prefixed with URL: or both.

scheme
: Specifies the value ldap for a non-SSL connection and ldaps for an SSL connection.

host:port
: Specifies the location of the LDAP server. The host specification can be a DNS resource name (for example, dcesec4.endicott.ibm.com), a dotted-decimal IPv4 address (for example, 9.130.25.34), or a colon-separated IPv6 address enclosed in square brackets (for example, [1080::8:800:200C:417A]). The port, if specified, must be a decimal number from 1 - 65535. The port defaults to 389 for a non-SSL connection and 636 for an SSL connection.

dn Specifies the distinguished name (DN) for the request. The DN can be used as a filter when the ldap_server_locate() routine should be called to locate the LDAP server.

attributes
Consists of one or more comma-separated search attributes.

scope
Specifies the search scope and can be `base`, `one`, or `sub`.

filter
Specifies the search filter. The search filter as a null-terminated character string in UTF-8 or the local EBCDIC code page, as determined by the LDAP_OPT_UTF8_IO option for the LDAP handle. If you specify NULL or a zero-length string for this parameter, the search filter is set to "(objectClass=*)".

Search filters can be used in LDAP URLs and LDAPSRCH. Search filters are constructed as defined in *RFC 2254: String Representation of LDAP Search Filters*. The filter syntax is defined by the rules shown below.

Rules:

```
filter        =  "(" filtercomp ")"
filtercomp    =  and / or / not / item
and           =  "&" filterlist
or            =  "|" filterlist
not           =  "!" filter
filterlist    =  1*filter
item          =  simple / present / substring / extensible
simple        =  attr filtertype value
filtertype    =  equal / approx / greatereq / lesseq
equal         =  "="
approx        =  "~="
greatereq     =  ">="
lesseq        =  "<="
extensible    =  attr [":dn"] [":" matchingrule] ":=" value
                    / [":dn"]  ":" matchingrule  ":=" value
present       =  attr "=*"
substring     =  attr "=" [initial] any [final]
initial       =  value
any           =  "*" *(value "*")
final         =  value
attr          =  AttributeDescription as defined in RFC 2251
matchingrule  =  MatchingRuleId as defined in RFC 2251
value         =  AttributeValue as defined in RFC 2251
```

Values inside double quotation marks represent literal values. Items enclosed in square brackets are optional. Items separated by / represent a choice. The notation `1*filter` indicates one or more filters. Specifying `":dn"` as part of the extensible item indicates that the components of the distinguished name are to be included in the matching as well as the object attributes.

An error is returned if an extensible filter item is specified and the LDAP protocol version is not `LDAP_VERSION3`.

Leading and trailing whitespace characters are ignored. Embedded whitespace characters are allowed within an attribute value and are retained. Embedded whitespace characters are not allowed within any of the literals in the above rules. Quotation marks have no special meaning within a search filter and are treated as normal characters.

Filter control characters, such as "(", ")", "*", and "\", within an attribute value must be escaped using the format "*xx*", where *xx* is the hexadecimal representation of the ASCII value of the escaped character. For example, "*" would be represented as "\2a". UTF-8 characters can be represented as a sequence of escaped characters; for example, "(sn=Lu\c4\8di\c4\87)". The case of the hexadecimal characters is not important.

IETF RFC 2254 replaces IETF RFC 1960. However, RFC 1960 specified that filter control characters were escaped by preceding the escaped control character with a reverse slash. For example, "*" would be represented as "*" within an attribute value. In order to provide compatibility with applications written to RFC 1960, filter control characters can be escaped using either format.

Earlier levels of LDAP allowed the outer parentheses to be omitted from the filter. For compatibility, z/VM LDAP allows the outer parentheses to be omitted from the filter. For example, "mail=*"can be specified instead of "(mail=*)".

Examples: The following are some examples of filters:

- (mail=*)

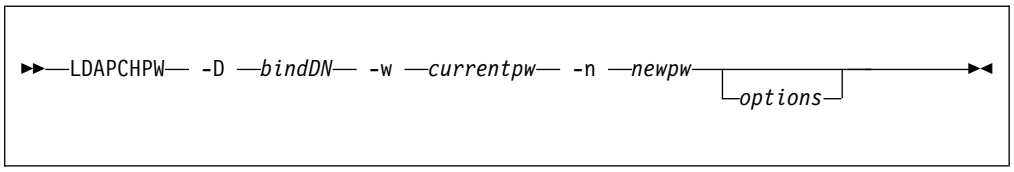
  This filter matches any entry with the mail attribute and does not match entries without the mail attribute.
- (mail=*@student.of.life.edu)

  This filter matches any entry whose mail attribute value ends with the string "@student.of.life.edu".
- (&(cn=Jane*)(sn=Doe)(!(uid=jdoe)))

  This filter matches any entry whose cn attribute value starts with Jane and whose sn attribute value is Doe and whose uid attribute value is not jdoe.

LDAPCHPW (ldapchangepwd Utility)

Format



Purpose

LDAPCHPW (**ldapchangepwd** utility) allows the **userPassword** attribute value to be changed for the specified entry.

The LDAPCHPW utility opens a connection to an LDAP server, binds, and sends modify password requests to the LDAP server. The input consists of a distinguished name (DN), a current password value, and a new password value. The current password value is deleted from the **userPassword** attribute values for the specified distinguished name and is replaced with the new password value.

Parameters

-D *bindDN*
Use *bindDN* to bind to the LDAP directory. The *bindDN* also identifies the DN

whose **userPassword** attribute value is to be changed. The *bindDN* parameter is required and should be a string-represented DN.

If the **-S** or **-m** option is equal to DIGEST-MD5 or CRAM-MD5, this option is the authorization DN that is used for making access checks.

-w *currentpw*
Use *currentpw* as the password for simple, CRAM-MD5, and DIGEST-MD5 authentication. This value also specifies the current **userPassword** attribute value that is being changed for the distinguished name (DN) specified by the **-D** option. This value is replaced by the new password value specified in the **-n** option. Specify **?** to prompt for the current password value. This option is required.

-n *newpw*
Specify the new **userPassword** attribute value for the distinguished name (DN) specified in the **-D** option. This value replaces the current password specified in the **-w** option. Specify **?** to prompt for the new password value. This option is required.

options
Table 23 shows the *options* you can use for the LDAPCHPW utility:

Table 23. LDAPCHPW options

Option	Description
-?	Print this text.
-d *debugLevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-g *realmName*	Specify the realm name to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.
-h *ldapHost*	Specify the hostname or IP address on which the LDAP server is running. The default is the local host.
-K *keyFile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the SSL_KEYRING environment variable with an associated name. This parameter is ignored if **-Z** is not specified.
-m *mechanism*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.
-N *keyFileDN*	Specify the label associated with the certificate in the SSL key database. This parameter is ignored if **-Z** is not specified
-p *ldapPort*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.

Table 23. LDAPCHPW options (continued)

Option	Description
-P *keyFilePW*	Specify either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form **file://** followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). This parameter is ignored if **-Z** is not specified.
-R	Do not automatically follow referrals.
-S *mechanism* or **-m** *mechanism*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. Specify `EXTERNAL` to indicate that a certificate (SASL external) bind is requested, `CRAM-MD5` to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or `DIGEST-MD5` to indicate a SASL digest hash bind is requested. The `EXTERNAL` method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is no default certificate in the key database file or a certificate other than the default needs to be used, use the **-N** option to specify the label of the certificate. The `CRAM-MD5` method requires a protocol level of 3. The **-D** or **-U** option must be specified. The `DIGEST-MD5` method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN. If neither **-m** nor **-S** is specified, a simple bind is performed.
-U *userName*	Specify the user name for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. This option is required if the **-S** or **-m** option is set to `DIGEST-MD5`.
-v	Use verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be **2** or **3**. The default is **3**.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyFile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyFilePW* option is required when the **-Z** option is specified and the key file specifies a file system key database file. Unless you want to use the default certificate in the key database file, use the **-N** option to specify the label of the certificate.

Examples

Examples of LDAPCHPW are:

- The following example changes the **userPassword** attribute value for entry cn=jon,o=ibm,c=us from a1b2c3d4 to wxyz9876 is:

```
ldapchpw -D "cn=jon,o=ibm,c=us" -w a1b2c3d4 -n wxyz9876
```

- The following example performs an EXTERNAL bind with the SSL client certificate, clientCert, in the /home/user/client.kdb SSL key database file. The authenticated user changes the **userPassword** value for entry cn=stanley,o=ibm,c=us from xyz123abc to abc321zyz:

```
ldapchpw -Z -m EXTERNAL -K /home/user/client.kdb -N clientCert -P secret
 -D "cn=stanley,o=ibm,c=us" -w xyz123abc -n abc321xyz
```

- The following example performs a simple bind to prompt for the current and new **userPassword** attribute values for entry cn=yvonne,o=ibm,c=us. At the prompts, the user enters the current and new password values in a non-echoed manner for user cn=yvonne,o=ibm,c=us.

```
ldapchpw -D "cn=yvonne,o=ibm,c=us" -w ? -n ?
Enter current password ==>
Enter new password ==>
```

Usage

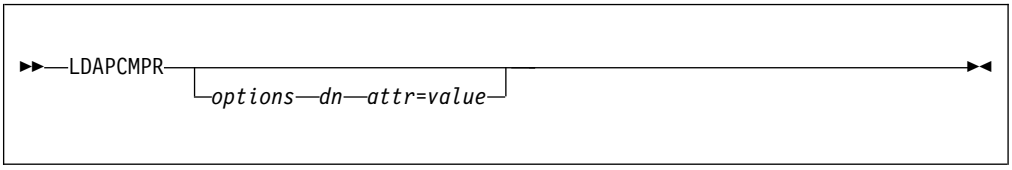
1. The LDAP_DEBUG environment variable can be used to set the debug level. For more information on specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
2. You can specify an LDAP URL for *ldaphost* on the **-h** parameter. For information about the syntax of LDAP URLs, see "LDAP URLs and LDAP Filters" on page 144.
3. The authenticating user must have the appropriate permissions to update the **userPassword** attribute for the distinguished name specified in the **-D** option.
4. The routine used to prompt for the password returns at most 255 characters, truncating any additional characters. If the length of the specified password is greater than 255, the password value must be specified on the **-w** option.

Return codes

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written. If the errors are caused by the password policy requirements not being met, the **Effective password policy** extended operation is invoked and the effective password policy entries and attributes values are written to standard output.

LDAPCMPR (ldapcompare Utility)

Format



Purpose

LDAPCMPR (**ldapcompare** utility) opens a connection to an LDAP server, binds, and does one or more compares for an attribute value in an entry. The input consists of a distinguished name (DN) and an attribute type and value to compare. For each set of input, a comparison is performed for the specified attribute in the

entry with that DN. If the DN and attribute type and value are not provided, the input is read from standard input or from *file* if the **-f** option is used, and two lines of input are read for each comparison. The first line contains the DN and the second line contains the attribute type and value.

Operands

options
Table 24 shows the *options* you can use for LDAPCMPR:

Table 24. LDAPCMPR options

Option	Description
-?	Print this text.
-c	Continuous operation mode. Errors are reported, but LDAPCMPR continues with comparisons. The default is to exit after reporting an error.
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a `NULL` string. If the **-S** or **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN that is used for making access checks. This directive is optional when used in this manner.
-f *file*	Read the compare input from *file* instead of from standard input or the command line (by specifying *dn* and *attr=value*). An LDAP compare is performed for every set of two lines in the file. The first line in the set specifies the DN of the entry to compare. The second line contains the *attr=value* specification, indicating the attribute and value to compare. For more information on specifying files, see "Specifying a value for a filename" on page 139.
-g *realmName*	Specify the realm name to use when doing a `DIGEST-MD5` bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a `DIGEST-MD5` challenge; otherwise, it is optional.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host.
-K *keyfile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name. SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, then the current directory is assumed to contain the file. For information on SSL key databases, see "SSL/TLS information for LDAP utilities" on page 139. This parameter is ignored if **-Z** is not specified.
-m *method*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.

Table 24. LDAPCMPR options (continued)

Option	Description
-n	Show what would be done, but do not actually compare entries. Useful for debugging in conjunction with **-v**.
-N *keyfilelabel*	Specify the label associated with the key in the SSL key database. This parameter is ignored if **-Z** is not specified.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-P *keyfilepw*	Specify either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form **file://** followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). This parameter is ignored if **-Z** is not specified.
-R	Do not automatically follow referrals.
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default method is `SIMPLE`. You can also specify `EXTERNAL` to indicate that a certificate (SASL external) bind is requested, `CRAM-MD5` to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or `DIGEST-MD5` to indicate a SASL digest hash bind is requested. The `EXTERNAL` method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate to be used. The `CRAM-MD5` method requires a protocol level of 3. The **-D** or **-U** option must be specified. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The `DIGEST-MD5` method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN.
-U *userName*	Specify the user name for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The **-U** option is required if the **-S** or **-m** option is set to `DIGEST-MD5`.
-v	Use verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be **2** or **3**. The default is **3**.
-w *passwd*	Use *passwd* as the password for simple, `CRAM-MD5`, and `DIGEST-MD5` authentication. The default is a `NULL` string.

Table 24. LDAPCMPR options (continued)

Option	Description
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyfile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyfilepw* option is required when the **-Z** option is specified and the key file specifies a file system key database file. The **-N** *keyfilelabel* option must be specified if you wish to use a certificate that is different than the default specified in the key database.

dn Specify the DN of the entry to compare.

attr=value
Specify the attribute type and the value to compare. An error is returned if the entry does not contain the attribute to be compared.

Return Codes

Exit status is `0` if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

Usage Notes

1. If no *dn* and *attr=value* arguments are provided and the **-f** option is not used, the LDAPCMPR command waits to read a list of DNs and attribute types and values from standard input. To break out of the wait, type HX and press the Enter key.
2. The `LDAP_DEBUG` environment variable can be used to set the debug level. For more information on specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
3. You can specify an LDAP URL for *ldaphost* on the **-h** parameter. For information about the syntax of LDAP URLs, see "LDAP URLs and LDAP Filters" on page 144.
4. For information about SSL/TLS, see "SSL/TLS information for LDAP utilities" on page 139.

Example

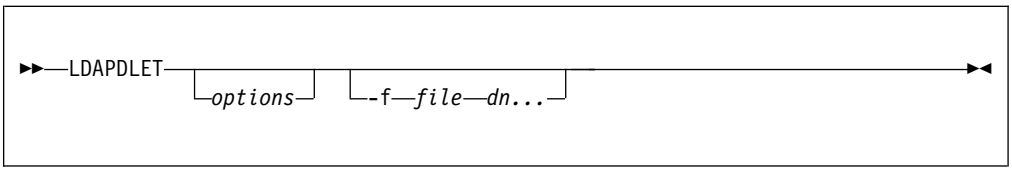
Following is an LDAPCMPR example:

- The following command compares the `sn` attribute within the entry named `cn=Compare Me, o=My Company, c=US`. The command returns true if the `sn` attribute value is `Smith` and false if it is not.

```
ldapcmpr "cn=Compare Me, o=My Company, c=US" sn=Smith
```

LDAPDLET (ldapdelete utility)

Format



Purpose

The LDAPDLET command (**ldapdelete** utility) opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more *dn* arguments are provided, entries with those DNs are deleted. If no *dn* arguments are provided, the input is read from standard input or from *file* if the **-f** flag is used. Each line of input contains the DN of an entry to be deleted. Each entry to be deleted must be a *leaf* entry (an entry with no subordinate entries) or it must become a leaf entry when the previously specified entries are deleted.

Operands

options

Table 25 shows the *options* you can use for LDAPDLET:

Table 25. LDAPDLET options

Option	Description
-?	Print this text.
-c	Continuous operation mode. Errors are reported, but LDAPDLET continues with deletions. The default is to exit after reporting an error.
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a `NULL` string. If the **-S** or **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN that is used for making access checks. This directive is optional when used in this manner.
-f *file*	Read a series of lines from *file*, performing one LDAP delete for the DN on each line. For more information on specifying files, see "Specifying a value for a filename" on page 139.
-g *realmName*	Specify the realm name to use when doing a `DIGEST-MD5` bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a `DIGEST-MD5` challenge; otherwise, it is optional.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host.

Table 25. LDAPDLET options (continued)

Option	Description
-K *keyfile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name. SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, then the current directory is assumed to contain the file. For information on SSL key databases, see "SSL/TLS information for LDAP utilities" on page 139. This parameter is ignored if **-Z** is not specified.
-m *method*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.
-n	Show what would be done, but do not actually delete entries. Useful for debugging in conjunction with **-v**.
-N *keyfilelabel*	Specify the label associated with the key in the SSL key database. This parameter is ignored if **-Z** is not specified.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-P *keyfilepw*	Specify either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form **file://** followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). This parameter is ignored if **-Z** is not specified.
-R	Do not automatically follow referrals.
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default method is `SIMPLE`. `EXTERNAL` to indicate that a certificate (SASL external) bind is requested, `CRAM-MD5` to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or `DIGEST-MD5` to indicate a SASL digest hash bind is requested. The `EXTERNAL` method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate to be used. The `CRAM-MD5` method requires a protocol level of 3. The **-D** or **-U** option must be specified. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli® Directory Server. The `DIGEST-MD5` method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN.

Table 25. LDAPDLET options (continued)

Option	Description
-U *userName*	Specify the user name for CRAM-MD5 or DIGEST-MD5 binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. Do not specify the **-U** option for a CRAM-MD5 bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The **-U** option is required if the **-S** or **-m** option is set to DIGEST-MD5.
-v	Use verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be **2** or **3**. The default is **3**.
-w *passwd*	Use *passwd* as the password for simple, CRAM-MD5, and DIGEST-MD5 authentication. The default is a NULL string.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyfile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyfilepw* option is required when the **-Z** option is specified and the key file specifies a file system key database file. The **-N** *keyfilelabel* option must be specified if you wish to use a certificate that is different than the default specified in the key database.

dn Specify distinguished name (DN) of an entry to delete. You can specify one or more *dn* arguments. Each *dn* should be a string-represented DN.

Return Codes

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

Usage Notes

1. If no *dn* arguments are provided and the **-f** option is not specified, the LDAPDLET command waits to read a list of DNs from standard input. To break out of the wait, type HX and press the Enter key.
2. The LDAP_DEBUG environment variable can be used to set the debug level. For more information on specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
3. You can specify an LDAP URL for *ldaphost* on the **-h** parameter. For information about the syntax of LDAP URLs, see "LDAP URLs and LDAP Filters" on page 144.
4. For information about SSL/TLS, see "SSL/TLS information for LDAP utilities" on page 139.

Example

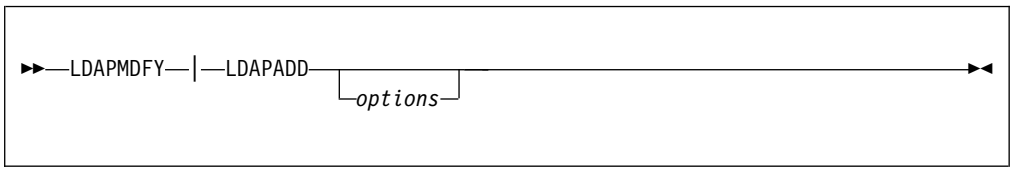
Following is an LDAPDLET example:

- The following command attempts to delete the entry named with commonName Delete Me directly below My Company organizational entry. It might be necessary to supply a *binddn* and *passwd* for deletion to be allowed. (See the **-D** and **-w** options.)

```
ldapdlet "cn=Delete Me, o=My Company, c=US"
```

LDAPMDFY and LDAPADD (ldapmodify and ldapadd Utilities)

Format



Purpose

The LDAPADD (**ldapadd** utility) command is implemented as a renamed version of LDAPMDFY (**ldapmdfy** utility). When invoked as LDAPADD, the **-a** (add new entry) flag is turned on automatically.

LDAPMDFY (**ldapmdfy** utility) opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read from standard input (or an input file redirected to standard input) or from *file* through the use of the **-f** option.

Operands

options
Table 26 shows the options you can use for LDAPMDFY and LDAPADD:

Table 26. LDAPMDFY and LDAPADD options

Option	Description
-?	Print this text.
-a	Add new entries. The default for LDAPMDFY is to modify existing entries. If invoked as LDAPADD, this flag is always set.
-b	Assume that any values that start with a slash (/) are binary values and that the actual value is in a file whose path is specified in the place where values normally appear.
-c	Continuous operation mode. Errors are reported, but LDAPMDFY continues with modifications. The default is to exit after reporting an error.
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a `NULL` string. If the **-S** or **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN that is used for making access checks. This directive is optional when used in this manner.
-f *file*	Read the entry modification information from *file* instead of from standard input. For more information on specifying files, see "Specifying a value for a filename" on page 139.

Table 26. LDAPMDFY and LDAPADD options (continued)

Option	Description
-F	Force application of all changes regardless of the contents of input lines that begin with **replica:** (by default, **replica:** lines are compared against the LDAP server host and port in use to decide if a replication log record should actually be applied).
-g *realmName*	Specify the realm name to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host.
-K *keyfile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the SSL_KEYRING environment variable with an associated name. SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, the current directory is assumed to contain the file. For information on SSL key databases, see "SSL/TLS information for LDAP utilities" on page 139. This parameter is ignored if **-Z** is not specified.
-m *method*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.
-n	Show what would be done, but do not actually modify entries. Useful for debugging in conjunction with **-v**.
-N *keyfilelabel*	Specify the label associated with the key in the SSL key database. This parameter is ignored if **-Z** is not specified.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-P *keyfilepw*	Specify either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form **file://** followed immediately (no blanks) by the file system file specification (for example, file:///etc/ldap/sslstashfile). This parameter is ignored if **-Z** is not specified.
-r	Replace existing values by default.
-R	Do not automatically follow referrals.

Table 26. LDAPMDFY and LDAPADD options (continued)

Option	Description
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default method is SIMPLE. EXTERNAL to indicate that a certificate (SASL external) bind is requested, CRAM-MD5 to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or DIGEST-MD5 to indicate a SASL digest hash bind is requested. The EXTERNAL method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate to be used. The CRAM-MD5 method requires a protocol level of 3. The **-D** or **-U** option must be specified. Do not specify the **-U** option for a CRAM-MD5 bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The DIGEST-MD5 method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN.
-u on | off	Specify whether LDAPMDFY sends the SchemaReplaceByValueControl control to the server. This control indicates how a schema modify reacts to a replace modification. If set to off, a schema modification removes all current values for an attribute and replaces them with the set of values in the replace operation. If set to on, an attribute value is updated if some value in the replace operation has the same numeric object identifier as a value that already exists in the schema attribute. An attribute value is added if the numeric object identifier in the replace operation does not exist in the schema attribute. No attribute values are removed.
-U *userName*	Specify the user name for CRAM-MD5 or DIGEST-MD5 binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. Do not specify the **-U** option for a CRAM-MD5 bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The **-U** option is required if the **-S** or **-m** option is set to DIGEST-MD5.
-v	Use verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be 2 or 3. The default is 3.
-w *passwd*	Use *passwd* as the password for simple, CRAM-MD5, and DIGEST-MD5 authentication. The default is a NULL string.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyfile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyfilepw* option is required when the **-Z** option is specified and the key file specifies a file system key database file. The **-N** *keyfilelabel* option must be specified if you wish to use a certificate that is different than the default specified in the key database.

All other command line inputs result in a syntax error message, after which the correct syntax is displayed. If the same option is specified multiple times or if both **-m** and **-S** are specified, the last value specified is used.

LDAP Data Interchange Format (LDIF)

LDAP Data Interchange Format (LDIF) is a standard text format for representing LDAP objects and LDAP updates (add, modify, delete, modify DN). Files containing LDIF records are used to transfer data between directory servers or used as input by LDAP utilities such as **ldapadd** and **ldapmodify**.

LDIF content records are used to represent LDAP directory content and consist of a line identifying the object, followed by optional lines containing controls, which are then followed by lines containing the attribute-value pairs for the object. This type of file is used by the **ldapadd** and **ds2ldif** utilities. See *z/VM: TCP/IP Planning and Customization* for additional information about the **ds2ldif** utility.

LDIF change records are used to represent directory updates. These records consist of a line identifying the directory object, followed by lines describing the changes to the object. The changes include adding, deleting, renaming, or moving objects, and modifying existing objects.

The input styles for content and change records are:

- A standard LDIF style defined by RFC 2849: *The LDAP Data Interchange Format (LDIF)*
- A non-standard "modify style"

Use of the standard LDIF style is recommended; the non-standard style is documented later for use with older tools that produce or use that style.

Input styles: The **ldapmodify** and **ldapadd** commands accept two forms of input. The type of input is determined by the format of the first input line supplied to **ldapmodify** or **ldapadd**.

The first line of input to the **ldapmodify** or **ldapadd** command must denote the distinguished name of a directory entry to add or modify. This input line must be of the form:

```
dn:distinguished_name
```

or

```
distinguished_name
```

where **dn:** is a literal string and *distinguished_name* is the distinguished name of the directory entry to modify (or add). If **dn:** is found, the input style is set to RFC 2849 LDIF style. If it is not found, the input style is set to "modify style".

Note:

1. The **ldapadd** command is equivalent to invoking the **ldapmodify -a** command.
2. The **ldapmodify** and **ldapadd** utilities do not support base64 encoded distinguished names.

RFC 2849 LDIF input

When using RFC 2849 LDIF input, attribute types and values are delimited by a single colon (:) or a double colon (::). Furthermore, individual changes to attribute values are delimited with a **changetype:** input line. The general form of input lines for RFC 2849 LDIF is:

```
change_record
<blank line>
change_record
<blank line>
 .
 .
 .
```

In RFC 2849 LDIF input:

1. A comment line is a line that begins with a number sign (#) in column 1. Comment lines are ignored.
2. A continuation line is a line that begins with a space in column 1. The rest of the continuation line, starting in column 2, is appended to the previous line.
3. Ensure that there are no extraneous spaces or characters at the end of a line. Even if not viewable in an editor, these characters are part of the modify input and can produce unexpected errors or unusable data.

An input file in RFC 2849 LDIF style consists of one or more *change_record* sets of lines that are separated by one or more blank lines. Each *change_record* has the following form:

```
dn:distinguished_name
[control:control_oid[true|false]
[::control_value]]
[changetype:{modify|add|modrdn|delete}]
{change_clause
.
.
.
}
```

A *change_record* consists of a line indicating the distinguished name of the entry directory to be modified, one or more optional lines indicating controls to be sent to the server on the modification, an optional line indicating the type of modification to be performed against the directory entry, and one or more *change_clause* sets of lines.

If one or more `control:` lines are present, the *control_oid* indicates the OID of the control, `true` or `false` may optionally be specified to indicate the criticality of the control (defaults to false if not specified), and an optional *control_value* can be specified. The *control_value* is expected to be in base64 format. This format is an encoding that represents every three binary bytes with four text characters. See the **base64encode()** function in `LINE64 CSAMPLE` for an implementation of base64 encoding.

The control lines in the RFC 2849 LDIF input style provide a way to apply certain controls to an individual entry rather than using a command line option. For example, the **-M**, **-L**, and **-k** command line options send the controls you want for all entries in the RFC 2849 LDIF input style. Any acceptable server control can be specified on the control lines. If the same control is specified multiple times for an entry, the client sends multiple controls for the entry to the server. This can occur if

the control is specified using a command line option and on a control line for the entry, or on multiple control lines for the entry.

If the **changetype:** line is omitted, the change type is assumed to be **modify** unless the command invocation was **ldapmodify -a** or **ldapadd**, in which case the **changetype** is assumed to be **add**.

When the change type is **modify**, each *change_clause* is defined as a set of lines of the form:

```
add:x
{attrtype}{sep}{value}
.
.
.-
```

or

```
replace:x
{attrtype}{sep}{value}
.
.
.
-
```

or

```
delete:{attrtype}
[{attrtype}{sep}{value}]
.
.
.
-
```

or

```
{attrtype}{sep}{value}
.
.
.
```

Specifying **replace** replaces all existing values for the attribute with the specified set of attribute values except when modifying a schema entry by using the **-u** option with `SchemaReplaceByValueControl` enabled. (See the description of the **-u** option in Table 26 on page 156.) Specifying **add** adds to the existing set of attribute values. Specifying **delete** without any attribute-value pair records removes all the values for the specified attribute. Specifying **delete** followed by one or more attribute-value pair records removes only those values specified in the attribute-value pair records.

If an **add:x**, **replace:x**, or **delete:***attrtype* line (a change indicator) is specified, a line containing a hyphen (-) is expected as a closing delimiter for the changes. Attribute-value pairs are expected on the input lines that are found between the change indicator and hyphen line. If the change indicator line is omitted, the change is assumed to be **add** for the attribute values specified. However, if the **-r** option is specified on **ldapmodify**, the *change_clause* is assumed to be **replace**. The separator, *sep*, can be either a single colon (:) or a double colon (: :). A single colon (:) is used as a separator when *value* contains printable characters while a double colon (: :) is used as a separator when *value* contains non-printable characters or begins with a space. Any white space characters between the separator and the attribute value are ignored. If a double colon is used as the separator, the input is expected to be in base64-encoded format. This format is an encoding that

represents every three binary bytes with four text characters. See the **base64encode()** function in `LINE64 CSAMPLE` for an implementation of this encoding.

Multiple attribute values are specified by using multiple {*attrtype*}{*sep*}{*value*} specifications.

Note: RFC 2849 indicates that LDIF file parsers should support the `file:// URL` format on a value to indicate that the contents of the referenced file are to be included verbatim in the integrated input of the LDIF file. However, the z/VM LDAP client LDIF parser does not support this specification. Also, the z/VM LDAP client LDIF parser does not support language or syntax tags on *attrtype*.

When the change type is **add**, each *change_clause* is defined as a set of lines of the form:

{*attrtype*}{*sep*}{*value*}

As with change type of **modify**, the separator, *sep*, can be either a single colon (:) or a double colon (: :). A single colon (:) is used as a separator when *value* contains printable characters while a double colon (: :) is used as a separator when *value* contains non-printable characters or begins with a space. Any white space characters between the separator and the attribute value are ignored. Attribute values can be continued across multiple lines by using a single space character as the first character of the next line of input. If a double colon is used as the separator, the input is expected to be in base64-encoded format.

When the change type is **modrdn**, each *change_clause* is defined as a set of lines of the form:

newrdn:*value***deleteoldrdn:{0|1}**

These are the parameters you can specify on a modify RDN LDAP operation. The value for the **newrdn** setting is the new RDN to be used when performing the modify RDN operation. Specify 0 for the value of the **deleteoldrdn** setting in order to save the attribute in the old RDN and specify 1 to remove the attribute values in the old RDN. You cannot use **ldapmodify** to move an entry under a new superior DN, instead, use "LDAPMRDN (ldapmodrdn Utility)" on page 169 with the **-s** option.

When the change type is **delete**, no *change_clause* is specified.

RFC 2849 LDIF Style examples

Here are some examples of valid input for the **ldapmodify** command using RFC 2849 LDIF style.

Adding a new entry: The following example adds a new entry into the directory using name `cn=Tim Doe, ou=Your Department, o=Your Company, c=US`, assuming **ldapadd** or **ldapmodify -a** is invoked:

```
dn:cn=Tim Doe, ou=Your Department, o=Your Company, c=US
changetype:add
cn: Tim Doe
sn: Doe
objectclass: organizationalperson
objectclass: person
objectclass: top
```

The following example sends the Server Administration Control (OID 1.3.18.0.2.10.15) and the Do Not Replicate Control (OID 1.3.18.0.2.10.23) and adds a new entry into the directory by using name cn=Tim Doe, ou=Your Department, o=Your Company, c=US, assuming **ldapadd** or **ldapmodify -a** is invoked:

```
dn:cn=Tim Doe, ou=Your Department, o=Your Company, c=US
control: 1.3.18.0.2.10.15
control: 1.3.18.0.2.10.23
changetype:add
cn: Tim Doe
sn: Doe
objectclass: organizationalperson
objectclass: person
objectclass: top
```

The following example adds a new entry that contains a binary user certificate that is base64-encoded in the userCertificate attribute into the directory by using name cn=John Doe, ou=Your Department, o=Your Company, c=US, assuming **ldapadd** or **ldapmodify -a** is invoked:

```
dn: cn=John Doe, ou=Your Department, o=Your Company, c=US
changetype:add
cn: John Doe
sn: Doe
usercertificate:: MIICNjCCAZ+gAwIBAgIBADANBgkqhkiG9w0BAQUFADAvMQswCQYDVQQGEwJ
1czEMMAoGA1UEChMDaWJtMRIwEAYDVQQDEwlyMTNzZXJ2ZXIwHhcNMTAwNjE1MDQwMDAwWhcNMjE
wMTAxMDM1OTU5WjAvMQswCQYDVQQGEwJ1czEMMAoGA1UEChMDaWJtMRIwEAYDVQQDEwlyMTNzZXJ
2ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMVgV8f3IAZEZ5/h3R2Iy7h4LSHbhsj4diH
iHPIpTRtqJD5d42z2Z4gG9oUzqfYLyZSPoAVlDwVbufZVVvBeiDo7Bgm+1nj4/YYWCpnCkETmriB
bVDJBoaF8W9xxHs38F6LVuJniDMp0VT9lDcqH3RNWgIcDqKurQm2uTHNDs6OtAgMBAAGjYjBgMD8
GCWCGSAGG+EIBDQQyFjBHZW5lcmF0ZWQgYnkgdGhlIFNlY3VyaXR5IFNlcnZlciBmb3Igei9PUyA
oUkFDRikwHQYDVR0OBBYEFLSjexfulLGxaf4xDvXV4Qhocv/JMA0GCSqGSIb3DQEBBQUAA4GBAIz
fNvc3kWSINsVNexPANbUG9i7SR/79B++pBszHwlKsDqCcB/Sa45yIIxni6cCnLFAoKQO76wFXAnC
Y4QDAxxukBdkiBjus0dQ4vfUDU2b5w+7F8mnvzNuHqvqBhk5DaMPbctcBl2E8lJkn3OwAk6VU+b5
 F6YJ3NT6y6SNDVk2q
objectclass: inetOrgPerson
objectclass: person
objectclass: top
```

Adding Attribute types: The following example sends the Server Administration Control (OID 1.3.18.0.2.10.15) and adds two new attribute types to the existing entry. Note the registeredaddress attribute is assigned two values:

```
dn:cn=Tim Doe, ou=Your Department, o=Your Company, c=US
control: 1.3.18.0.2.10.15
changetype:modify
add:x
telephonenumber: 888 555 1234
registeredaddress: td@yourcompany.com
registeredaddress: ttd@yourcompany.com
-
```

Changing the Entry name: The following example changes the name of the existing entry to cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US. The old RDN, cn=Tim Doe, is retained as an additional attribute value of the cn attribute. The new RDN, cn=Tim Tom Doe, is added automatically by the LDAP server to the values of the cn attribute in the entry:

```
dn: cn=Tim Doe, ou=Your Department, o=Your Company, c=US
changetype:modrdn
newrdn: cn=Tim Tom Doe
deleteoldrdn: 0
```

Replacing Attribute values: The following example replaces the attribute values for the `telephonenumber` and `registeredaddress` attributes with the specified attribute values:

```
dn: cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US
changetype:modify
replace:x
telephonenumber: 888 555 4321
registeredaddress: tim@yourcompany.com
registeredaddress: timtd@yourcompany.com
-
```

Deleting and adding Attributes: The following example deletes the `telephonenumber` attribute, deletes a single `registeredaddress` attribute value, and adds a description attribute:

```
dn:cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US
changetype:modify
add:x
description: This is a very long attribute
value that is continued on a second line.
Note the spacing at the beginning of the
continued lines in order to signify that
the line is continued.
-
delete: telephonenumber
-
delete: registeredaddress
registeredaddress: tim@yourcompany.com
-
```

Modifying Multiple entries: The following example adds the `postalCode` attribute and replaces the `description` attribute in the directory entry with name `cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US` and adds a new directory entry with name `cn=Ken Smith, ou=Your Department, o=Your Company, c=US`.

Note: A line containing only a dash is used to separate different types of changes within an entry and a blank line (a line containing no characters) is used to separate the changes to different entries.

```
dn: cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US
changetype: modify
add: x
postalcode: 12345
-
replace: x
description: This is a short description.
-
dn: cn=Ken Smith, ou=Your Department, o=Your Company, c=US
changetype: add
cn: Ken Smith
sn: Smith
objectclass: organizationalperson
```

Deleting an entry: The following example deletes the directory entry with name `cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US`:

```
dn:cn=Tim Tom Doe, ou=Your Department, o=Your Company, c=US
changetype:delete
```

Modify Style

The "modify style" of input to the **ldapmodify** or **ldapadd** commands is not as flexible as the RFC 2849 LDIF style. However, it is sometimes easier to use than the LDIF style.

When using modify style input, attribute types and values are delimited by an equal sign (=). The general form of input lines for modify style is:

```
change_record
<blank line>change_record
<blank line>
.
.
.
```

In modify style input:

1. A comment line is a line that begins with a number sign (#) in column 1. Comment lines are ignored.
2. A line can be continued by specifying a backslash (\) as the last character of the line. If a line is continued, the backslash character is removed and the succeeding line is appended directly after the character preceding the backslash character.

An input file in modify style consists of one or more *change_record* sets of lines separated by a single blank line. Each *change_record* has the following form:

```
distinguished_name
[+|-]{attrtype} = {value_line1[\
value_line2[\
...value_lineN]]}
.
.
.
```

Therefore, a *change_record* consists of a line indicating the distinguished name of the directory entry to be modified along with one or more attribute modification lines. Each attribute modification line consists of an optional add or delete indicator (+ or −), an attribute type, and an attribute value. If a plus sign (+) is specified, the modification type is set to **add**. If a hyphen (−) is specified, the modification type is set to **delete**. For a delete modification, the equal sign (=) and *value* should be omitted to remove an entire attribute. If the add or delete indicator is not specified, the modification type is set to **add** unless the **-r** option is used, in which case the modification type is set to **replace**. Any leading or trailing white space characters are removed from attribute values. If trailing white space characters are required for attribute values, the RFC 2849 LDIF style of input must be used. The new-line character at the end of the input line is not retained as part of the attribute value.

Multiple attribute values are specified by using multiple *attrtype=value* specifications.

Modify style examples

Here are some examples of valid input for the **ldapmodify** command by using modify style.

Adding a new entry: The following example adds a new entry into the directory by using name `cn=Tim Doe, ou=Your Department, o=Your Company, c=US`:

```
cn=Tim Doe, ou=Your Department, o=Your Company, c=US
cn=Tim Doe
sn=Doe
objectclass=organizationalperson
objectclass=person
objectclass=top
```

Adding a new attribute type: The following example adds two new attribute types to the existing entry. Note the `registeredaddress` attribute is assigned two values:

```
cn=Tim Doe, ou=Your Department, o=Your Company, c=US
+telephonenumber=888 555 1234
+registeredaddress=td@yourcompany.com
+registeredaddress=ttd@yourcompany.com
```

Replacing attribute values: Assuming that the command invocation was:

```
ldapmodify -r ...
```

The following example replaces the attribute values for the `telephonenumber` and `registeredaddress` attributes with the specified attribute values. If the **-r** command line option was not specified, the attribute values are added to the existing set of attribute values.

```
cn=Tim Doe, ou=Your Department, o=Your Company, c=US
telephonenumber=888 555 4321
registeredaddress: tim@yourcompany.com
registeredaddress: timtd@yourcompany.com
```

Deleting an attribute type: The following example deletes a single `registeredaddress` attribute value from the existing entry.

```
cn=Tim Doe, ou=Your Department, o=Your Company, c=US
-registeredaddress=tim@yourcompany.com
```

Adding an attribute: The following example adds a description attribute. The description attribute value spans multiple lines:

```
cn=Tim Doe, ou=Your Department, o=Your Company, c=US
+description=This is a very long attribute \
value that is continued on a second line. \
Note the backslash at the end of the line to \
be continued in order to signify that \
the line is continued.
```

Changing the numeric object identifier: A special input file is required to change the numeric object identifier (OID) of an attribute or an object class in the z/VM LDAP server schema. This input file must contain a delete of the existing attribute or object class (with the old OID) followed by an add of the new version of the attribute or object class (with the new OID). The value for NAME within the attribute or object class must be identical in the delete and add modifications. When using the z/VM LDAP Server, noncritical values (such as DESC) can be changed in the new version but critical values (such as the SYNTAX or the MUST and MAY lists) must be the same as in the existing attribute or object class. The deletion and addition must be the only modifications made to the schema in that operation.

For example, to change the OID for the `userHomeAddr` attribute from `1.3.21.7777` to `2.5.44.3.9999` in the schema, the input file for **ldapmodify** should contain:

```
cn=schema
-attributetypes=( 1.3.21.7777 NAME 'userHomeAddr' DESC 'The home address' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
+attributetypes=( 2.5.44.3.9999 NAME 'userHomeAddr' DESC 'The home address' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
```

Examples

Following are some **ldapmodify** and **ldapadd** examples. It might be necessary to supply a *bindDN* and *passwd* for modify to be allowed.

1. Assume that the /tmp/entrymods file exists and has the following contents:

```
dn: cn=Modify Me, o=My Company, c=US
changetype: modify
replace: mail
mail: modme@MyCompany.com
-
add: title
title: Vice President
-
add: jpegPhoto
jpegPhoto: /tmp/modme.jpeg
-
delete: description
-
```

   The following command replaces the contents of the Modify Me entry's mail attribute with the value modme@MyCompany.com, adds a title of Vice President, adds the contents of the file /tmp/modme.jpeg as the jpegPhoto value, and completely removes the description attribute.

```
ldapmodify -b -r -f /tmp/entrymods
```

   The same modifications as above can be performed by using the older **ldapmodify** input format:

```
cn=Modify Me, o=My Company, c=US
mail=modme@MyCompany.com
+title=Vice President
+jpegPhoto=/tmp/modme.jpeg
-description
```

2. Assume that the /tmp/certuser file exists and has the following contents:

```
dn: cn=Karen Smith, o=My Company, c=US
objectclass: inetorgperson
cn: Karen Smith
sn: Smith
userpassword: secret
usercertificate: //'USER.CERTDER'
```

   The following command adds a new entry for Karen Smith by using the values from the /tmp/certuser file. The usercertificate value is obtained from the binary data set USER.CERTDER.

```
ldapadd -b -f /tmp/certuser
```

3. Assume that the /tmp/newentry file exists and has the following contents:

```
dn: cn=Joe Smith, o=My Company, c=US
objectClass: person
cn: Joseph Smith
cn: Joe Smith
sn: Smith
title: Manager
mail: jsmith@jsmith.MyCompany.com
uid: jsmith
```

   The following command adds a new entry for Joe Smith by using the values from the /tmp/newentry file.

```
ldapadd -f /tmp/newentry
```

4. Assume that the /tmp/newentry file exists and has the following contents:

```
dn: cn=Joe Smith, o=My Company, c=US
changetype: delete
```

   The following command removes Joe Smith's entry.

```
ldapmodify -f /tmp/newentry
```

5. Assume that hostA contains the referral object:

```
dn: o=ABC,c=US
ref: ldap://hostB:390/o=ABC,c=US
objectclass: referral
```

and hostB contains the organization object:

```
dn: o=ABC,c=US
o: ABC
objectclass: organization
telephoneNumber: 123-4567
```

and the /tmp/refmods file has the following contents

```
dn: o=ABC,c=US
changetype: modify
replace: ref
ref: ldap://hostB:391/o=ABC,c=US
-
```

and the /tmp/ABCmods file has the following contents:

```
dn: o=ABC,c=US
changetype: modify
add: telephoneNumber
telephoneNumber: 123-1111
-
```

The following command replaces the ref attribute value of the referral object o=ABC,c=US in hostA, changing the TCP port address in the URL from 390 to 391.

```
ldapmodify -h hostA -r -M -f /tmp/refmods
```

The following command adds the telephoneNumber attribute value 123-1111 to o=ABC,c=US in hostB.

```
ldapmodify -h hostB -p 391 -f /tmp/ABCmods
```

6. Assume that the /tmp/schemamods file exists and has the following contents:

```
dn: cn=schema
-attributetypes=( 1.2.1 NAME 'attr1' DESC 'attribute type' \
  EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
+attributetypes=( 1.2.1 NAME 'attr1' DESC 'attribute type - obsoleted' OBSOLETE \
  EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
+attributetypes=( 1.2.2 NAME 'attr2' DESC 'new attribute type' \
  EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
+ibmattributetypes=( 1.2.2 ACCESS-CLASS normal )
-objectclasses=( 4.5.6 NAME 'oc1' DESC 'sample object class' STRUCTURAL MUST ( cn ) )
+objectclasses=( 4.5.6 NAME 'oc1' DESC 'sample object class' STRUCTURAL MUST ( cn ) MAY ( attr2 ) )
```

The following command obsoletes the attr1 attribute type definition by specifying the OBSOLETE keyword in the definition, adds the attr2 attribute type definition and the associated IBM attribute type information, and modifies the oc1 object class definition by adding the attr2 attribute type as a MAY attribute.

```
ldapmodify -f /tmp/schemamods
```

7. Assume that the /tmp/newentry file exists and has the following contents:

```
dn: racfid=u1,profiletype=user,sysplex=sysplexa
objectclass: racfuser
objectclass: racfbasecommon
racfid: u1
racfdefaultgroup: racfid=g1,profiletype=group,sysplex=sysplexa
racfconnectgroupUACC: read
racfconnectgroupauthority: jo
```

The following command creates a RACF user named u1, with join authority and update UACC in the group g1. It is assumed that the z/VM LDAP support for RACF access suffix is sysplex=sysplexa and that admin1 has the RACF authority to make this update to RACF

```
ldapadd -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd -f /tmp/newentry
```

8. Assume that the /tmp/modentry file contains the following attributes.

   Note: In the following LDIF, the x on the replace: x line is a placeholder for the attribute name and allows multiple attribute names and values to be replaced in a single operation.

   ```
   dn: racfid=u1,profiletype=user,sysplex=sysplexa
   changetype: modify
   replace: x
   racfattributes: OPERATIONS
   racfconnectgroupUACC: update
   ```

   The following command adds OPERATIONS to racfattributes and changes the racfconnectgroupUACC value to update.

   ```
   ldapmodify -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd -f /tmp/modentry
   ```

Return Codes

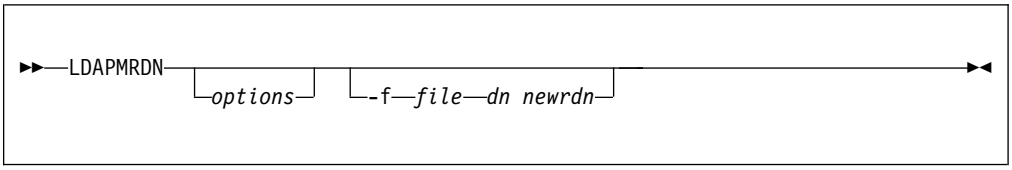
Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

Usage Notes®

1. The LDAP_DEBUG environment variable can be used to set the debug level. For more information about specifying the debug level by using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
2. You can specify an LDAP URL for *ldapHost* on the **-h** parameter. See *ldap_init()* for more information.
3. For information about SSL/TLS, see "SSL/TLS information for LDAP utilities" on page 139.

LDAPMRDN (ldapmodrdn Utility)

Format



Purpose

The LDAPMRDN command (**ldapmodrdn** utility) opens a connection to an LDAP server, binds, and modifies the DN of entries. The input consists of a distinguished name (DN) and a new relative distinguished name (RDN). The new RDN replaces the existing RDN in the entry specified by the DN. If no *dn* and *newrdn* arguments are provided, the input is read from standard input or from *file* if the **-f** flag is

used, and two lines are read for each rename. The first line contains the DN and the second line contains the new RDN. One or more blank lines must separate each DN and RDN pair.

The entries being renamed can be either leaf entries or non-leaf entries, and entire subtrees can be relocated in the directory with the **-s** option.

LDAPMRDN is not supported for RACF access on z/VM LDAP.

Operands

options
Table 27 shows the *options* you can use for LDAPMRDN:

Table 27. LDAPMRDN options

Option	Description
-?	Print this text.
-a	Sends an `IBMModifyDNRealignDNAttributesControl` control with the operation request. The control criticality is set to `TRUE`. There is no control value. For a description of this control, see *z/VM: TCP/IP LDAP Administration Guide*.
-c	Continuous operation mode. Errors are reported, but LDAPMRDN continues with modifications. The default is to exit after reporting an error.
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a `NULL` string. If the **-S** or **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN that is used for making access checks. This directive is optional when used in this manner.
-f *file*	Read the entry rename information from *file* instead of from standard input or the command line (by specifying *dn* and *newrdn*). Multiple pairs of *dn* and *newrdn* can be specified in the input file or standard input. The pairs must be separated by one or more blank lines. The *newSuperior* option cannot be included in *file*; this option is only accepted as a command-line option. If the *newSuperior* option (**-s**) is specified, each entry specified in the file will have its RDN updated and be moved under the new superior entry's DN. If the `IBMModifyDNRealignDNAttributesControl` option (**-a**) is specified, it is sent on each rename operation that is specified in the file. For more information on specifying files, see “Specifying a value for a filename” on page 139.
-g *realmName*	Specify the realm name to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host.

Table 27. LDAPMRDN options (continued)

Option	Description
-K *keyfile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name. SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, the current directory is assumed to contain the file. For information on SSL key databases, see "SSL/TLS information for LDAP utilities" on page 139. This parameter is ignored if **-Z** is not specified.
-l *seconds*	Sends an `IBMModifyDNTimelimitControl` control with the operation request, substituting *seconds* as the control value. The control criticality is set to `TRUE`. For a description of this control, see *z/VM: TCP/IP LDAP Administration Guide*.
-m *method*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.
-n	Show what would be done, but do not actually change entries. Useful for debugging in conjunction with **-v**.
-N *keyfilelabel*	Specify the label associated with the key in the SSL key database.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-P *keyfilepw*	Specify either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form **`file://`** followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). This parameter is ignored if **-Z** is not specified.
-r	Remove old RDN values from the entry. Default is to keep old values.
-R	Do not automatically follow referrals.
-s *newSuperior*	Specify the DN of the new superior entry under which the renamed entry will be relocated. The *newSuperior* argument can be the zero-length string (**-s** `""`), if the destination server accepts zero-length string *newSuperior* arguments on an LDAP `Modify DN` operation.

Table 27. LDAPMRDN options (continued)

Option	Description
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default method is `SIMPLE`. `EXTERNAL` to indicate that a certificate (SASL external) bind is requested, `CRAM-MD5` to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or `DIGEST-MD5` to indicate a SASL digest hash bind is requested. The `EXTERNAL` method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate is used. The `CRAM-MD5` method requires a protocol level of 3. The **-D** or **-U** option must be specified. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The `DIGEST-MD5` method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN.
-U *userName*	Specify the user name for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The **-U** option is required if the **-S** or **-m** option is set to `DIGEST-MD5`.
-v	Use verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be **2** or **3**. The default is **3**.
-w *passwd*	Use *passwd* as the password for simple, `CRAM-MD5`, and `DIGEST-MD5` authentication. The default is a `NULL` string.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyfile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyfilepw* option is required when the **-Z** option is specified and the key file specifies a file system key database file. The **-N** *keyfilelabel* option must be specified if you wish to use a certificate that is different than the default specified in the key database.

dn Specify the DN of the entry to change.

newrdn
Specify the new RDN for the entry.

Return Codes

Exit status is `0` if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

Usage Notes

1. The LDAP_DEBUG environment variable can be used to set the debug level. For more information on specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
2. You can specify an LDAP URL for *ldaphost* on the **-h** parameter. For information about the syntax of LDAP URLs, see "LDAP URLs and LDAP Filters" on page 144.
3. For clients using authenticated binds, the DNs in their identity mappings might change as a result of a Modify DN operation which is performed concurrently with their session to the server, and this might affect ACL processing which results in permission to access, or denial of access to, directory entries for which they previously were permitted or denied access. The resolution for this situation is to unbind and rebind, so that identity processing uses the latest DNs.
4. For information about SSL/TLS, see "SSL/TLS information for LDAP utilities" on page 139.

Examples

The following are LDAPMRDN examples.

1. Assume that the /tmp/entrymods file exists and has the following contents:

   ```
   cn=Modify Me, o=My Company, c=US
   cn=The New Me
   ```

   The following command changes the RDN from cn=Modify Me to cn=The New Me and removes the old RDN cn=Modify Me from the entry. The DN of the entry is cn=The New Me, o=My Company, c=US.

   ```
   ldapmrdn -r -f /tmp/entrymods
   ```

2. The following command is another way to accomplish the same change as Example 1. An IBMModifyDNTimelimitControl control accompanies the operation request, specifying a time limit of 30 seconds.

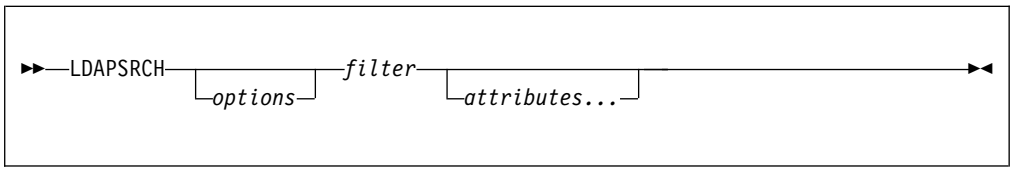
   ```
   ldapmrdn -r -l 30 "cn=Modify Me, o=My Company, c=US" "cn=The New Me"
   ```

3. The following command changes the RDN from cn=Modify Me to cn=The New Me and removes the old RDN cn=Modify Me from the entry. The renamed entry is relocated beneath the new superior entry o=Some Other Company, c=US. The DN of the entry is changed to cn=The New Me, o=Some Other Company, c=US. If the renamed entry is a non-leaf node, its subordinate entries are also moved and renamed to reflect their new locations in the directory hierarchy. An IBMModifyDNTimelimitControl control accompanies the operation request, specifying a time limit of 30 seconds, and an IBMModifyDNRealignDNAttributesControl control accompanies the operation request.

   ```
   ldapmrdn -l 30 -a -s "o=Some Other Company, c=US" \
    "cn=Modify Me, o=My Company, c=US" "cn=The New Me"
   ```

LDAPSRCH (ldapsearch Utility)

Format



Purpose

The LDAPSRCH command (**ldapsearch** utility) opens a connection to an LDAP server, binds, and performs a search using the specified filter. If LDAPSRCH finds one or more entries, the specified attributes are retrieved and the entries and values are printed to standard output.

Restriction: Use of the approximate filter (~=) is not supported on a z/VM LDAP Server. This filter is processed like an equality (=) filter.

Operands

options
Table 28 shows the *options* you can use for LDAPSRCH:

Table 28. LDAPSRCH options

Option	Description
-?	Print this text.
-a *deref*	Specify how alias dereferencing is done. The *deref* should be one of `never`, `always`, `search`, or `find` to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is to never dereference aliases.
-A	Retrieve attributes only (no values). This is useful when you just want to see if an attribute is present in an entry and are not interested in the specific values.
-b *searchbase*	Use *searchbase* as the starting point for the search instead of the default. If **-b** is not specified, this utility examines the `LDAP_BASEDN` environment variable for a *searchbase* definition. Set the `LDAP_BASEDN` environment variable using the `_CEE_ENVFILE` LE runtime environment variable. For more information, see *z/OS: XL C/C++ Programming Guide*. In CMS, environment variables are set with the GLOBALV SELECT command, specifying the CENV group. For information on the GLOBALV command, see *z/VM: CMS Commands and Utilities Reference*.
-B	Do not suppress display of non-printable values. This is useful when dealing with values that appear in alternate character sets such as ISO8859.1. This option is implied by the **-L** option.
-C	Do not suppress display of printable non-ASCII values (similar to the **-B** option). Values are displayed in the local code page. The `LANG` environment variable must be set appropriately in the shell so that the desired characters print. Note that the default `LANG` value of `C` causes the desired characters not to print.

Table 28. LDAPSRCH options (continued)

Option	Description
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server. For the possible values for *debuglevel*, see Table 22 on page 143. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a `NULL` string. If the **-S** or **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN that is used for making access checks. This directive is optional when used in this manner.
-f *file*	Read a series of lines from *file*, performing one LDAP search for each line. In this case, the *filter* given on the command line is treated as a pattern where the first occurrence of `%s` is replaced with a line from *file*. If *file* is a single hyphen (`-`) character, then the lines are read from standard input. For more information on specifying files, see "Specifying a value for a filename" on page 139.
-F *sep*	Use *sep* as the field separator between attribute names and values. The default separator is an equal sign (=), unless the **-L** flag has been specified, in which case this option is ignored.
-g *realmName*	Specify the realm name to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host.
-K *keyfile*	Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name. SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, the current directory is assumed to contain the file. For information on SSL key databases, see "SSL/TLS information for LDAP utilities" on page 139. This parameter is ignored if **-Z** is not specified.
-l *timelimit*	Limit the maximum wait time for the search request, overriding the value of `LDAP_OPT_TIMELIMIT` in the LDAP handle. Specify `NULL` for this parameter if there is no time limit for the request. Otherwise, set the *timeout* value to the maximum time in seconds. The LDAP server can also provide a limit on the search time. For information on the server's search time limit and how it interacts with the client time limit, see the documentation for your LDAP server. For the z/VM LDAP server, see the description of the `timeLimit` configuration file option in *z/VM: TCP/IP Planning and Customization*. The default time limit for the client, specified by a value of `0`, indicates that there is no client time limit and that the maximum number of seconds is limited only by the LDAP server limit.

Table 28. LDAPSRCH options (continued)

Option	Description
-L	Display search results in LDIF format. All characters in the LDIF format output are portable characters that are represented in the local code page. Binary attribute values are displayed in base64 encoded format. This option also turns on the **-B** option, and causes the **-F** option to be ignored. See "LDAPMDFY and LDAPADD (ldapmodify and ldapadd Utilities)" on page 156 for more information about the LDIF format. **Note:** LDIF output from the **ldapsearch** utility does not necessarily produce suitable data for backup and restore purposes. The **ds2ldif** utility should be considered as an alternative. The **ldapsearch** and **ds2ldif** utilities have several differences, including options to control the order of entries in the file (**ldapsearch** might not produce entries in correct hierarchical order), options to control entry contents (including operational attributes), and code pages used for portable characters in the output file. See ds2ldif utility in *z/VM: TCP/IP Planning and Customization* for more information.
-m *method*	See the description of the **-S** option.
-M	Manage referral objects as normal entries. This requires a protocol level of 3.
-n	Show what would be done, but do not actually perform the search. Useful for debugging in conjunction with **-v**.
-N *keyfiledn*	Specify the label associated with the key in the SSL key database.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-P *keyfilepw*	Specify either the key database file password or the file specification for an SSL password stash file. When the stash file is used, it must be in the form **file://** followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). This parameter is ignored if **-Z** is not specified.
-R	Do not automatically follow referrals.
-s *scope*	Specify the scope of the search. The *scope* should be one of `base`, `one`, or `sub` to specify a base object, one-level, or subtree search. The default is `sub`.
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default method is `SIMPLE`. `EXTERNAL` to indicate that a certificate (SASL external) bind is requested, `CRAM-MD5` to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or `DIGEST-MD5` to indicate a SASL digest hash bind is requested. The `EXTERNAL` method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate is used. The `CRAM-MD5` method requires a protocol level of 3. The **-D** or **-U** option must be specified. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The `DIGEST-MD5` method requires a protocol level of 3. The **-U** option must be specified. Optionally, the **-D** option can be used to specify the authorization DN.

Table 28. LDAPSRCH options (continued)

Option	Description
-t	Write retrieved values to a set of files in the `/tmp` directory, using file names similar to `/tmp/ldapsearch-objectclass-bbeFxQ`. This option assumes values are non-textual (binary), such as `jpegPhoto` or `audio`. There is no character set translation performed on the values.
-U *userName*	Specify the user name for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (for example, the *uid* attribute value) that is used to perform bind authentication. Do not specify the **-U** option for a `CRAM-MD5` bind to a non-z/OS implementation of the IBM Tivoli Directory Server. The **-U** option is required if the **-S** or **-m** option is set to `DIGEST-MD5`.
-v	Run in verbose mode, with many diagnostics written to standard output.
-V *version*	Specify the LDAP protocol level the client should use. The value for *version* can be **2** or **3**. The default is **3**.
-w *passwd*	Use *bind passwd* as the password for simple, CRAM-MD5, and DIGEST-MD5 authentication. The default is a `NULL` string.
-z *sizelimit*	Limit the number of entries that can be returned, overriding the value of `LDAP_OPT_SIZELIMIT` in the LDAP handle. A value of `0` indicates that there is no limit. The LDAP server can also provide a size limit on the number of entries returned. For information on the server's size limit and how it interacts with the client size limit, see the documentation for your LDAP server. For the z/VM LDAP server, see the description of the `sizeLimit` configuration file option in *z/VM: TCP/IP Planning and Customization*. The default size limit for the client, specified by a value of `0`, indicates that the maximum number of entries is limited only by the LDAP server limit.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** *keyfile* option or equivalent environment variable is required when the **-Z** option is specified. The **-P** *keyfilepw* option is required when the **-Z** option is specified and the key file specifies a file system key database file. The **-N** *keyfilelabel* option must be specified if you wish to use a certificate that is different than the default specified in the key database.

filter
: Specify an IETF RFC 1558-compliant LDAP search filter.

attributes
: Specify a space-separated list of attributes to retrieve. If no *attributes* list is given, all are retrieved.

All other command line inputs result in a syntax error message, after which the correct syntax is displayed. If the same option is specified multiple times or if both **-m** and **-S** are specified, the last value specified is used.

Output format

If one or more entries are found, each entry is written to standard output in the form:

```
Distinguished Name (DN)
attributename=value
attributename=value
attributename=value
...
```

Multiple entries are separated with a single blank line. If the **-F** option is used to specify a separator character, it is used instead of the equal sign (=). If the **-t** option is used, the name of a temporary file is used in place of the actual value. If the **-A** option is given, only the `attributename` part is written.

Examples

Following are some **ldapsearch** examples. Each example makes the assumption that the LDAP server is running on the local host and listening on the default LDAP port (389).

- The command:

  ```
  ldapsearch -b "o=IBM University,c=US" "cn=karen smith" cn telephoneNumber
  ```

  performs a subtree search using the search base "o=IBM University,c=US" for entries with a commonName of karen smith. The commonName and telephoneNumber values are retrieved and printed to standard output. The output might look something like this if two entries are found:

  ```
  cn=Karen G Smith, ou=College of Engineering, o=IBM University, c=US
  cn=Karen Smith
  cn=Karen Grace Smith
  cn=Karen G Smith
  telephoneNumber=+1 313 555-9489

  cn=Karen D Smith, ou=Information Technology Division, o=IBM University, c=US
  cn=Karen Smith
  cn=Karen Diane Smith
  cn=Karen D Smith
  telephoneNumber=+1 313 555-2277
  ```

- The command:

  ```
  ldapsearch -b "o=IBM University,c=US" -t "uid=kds" jpegPhoto audio
  ```

  Performs a subtree search using the search base "o=IBM University,c=US" for entries with user ID of kds. The jpegPhoto and audio values are retrieved and written to temporary files. The output might look like this if one entry with one value for each of the requested attributes is found:

  ```
  cn=Karen D Smith, ou=Information Technology Division, o=IBM University, c=US
  audio=/tmp/ldapsearch-audio-a19924
  jpegPhoto=/tmp/ldapsearch-jpegPhoto-a19924
  ```

- The command:

  ```
  ldapsearch -L -b "c=US" "(&(usercertificate=*)(objectclass=inetOrgPerson))" usercertificate cn
  ```

  Performs a subtree-level search at the c=US level for all users that have an objectclass value of inetOrgPerson and a userCertificate value. Search results are displayed in the LDIF format. The userCertificate attribute value is displayed in base64-encoded format because it is a binary value. The userCertificate and cn attribute values are retrieved and printed to standard output, resulting in output like the following:

```
dn: cn=John Doe, ou=Your Department, o=Your Company, c=US
cn: John Doe
usercertificate:: MIICNjCCAZ+gAwIBAgIBADANBgkqhkiG9w0BAQUFADAvMQswCQYDVQQGEwJ
 1czEMMAoGA1UEChMDaWJtMRIwEAYDVQQDEwlyMTNzZXJ2ZXIwHhcNMTAwNjE1MDQwMDAwWhcNMjE
 wMTAxMDM1OTU5WjAvMQswCQYDVQQGEwJ1czEMMAoGA1UEChMDaWJtMRIwEAYDVQQDEwlyMTNzZXJ
 2ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMVgV8f3IAZEZ5/h3R2Iy7h4LSHbhsj4diH
 iHPIpTRtqJD5d42z2Z4gG9oUzqfYLyZSPoAVlDwVbufZVVvBeiDo7Bgm+1nj4/YYWCpnCkETmriB
 bVDJBoaF8W9xxHs38F6LVuJniDMp0VT9lDcqH3RNWgIcDqKurQm2uTHNDs6OtAgMBAAGjYjBgMD8
 GCWCGSAGG+EIBDQQyFjBHZW5lcmF0ZWQgYnkgdGhlIFNlY3VyaXR5IFNlcnZlciBmb3Igei9PUyA
 oUkFDRikwHQYDVR0OBBYEFLSjexfulLGxaf4xDvXV4Qhocv/JMA0GCSqGSIb3DQEBBQUAA4GBAIz
 fNvc3kWSINsVNexPANbUG9i7SR/79B++pBszHwlKsDqCcB/Sa45yIIxni6cCnLFAoKQO76wFXAnC
 Y4QDAxxukBdkiBjus0dQ4vfUDU2b5w+7F8mnvzNuHqvqBhk5DaMPbctcBl2E8lJkn3OwAk6VU+b5
 F6YJ3NT6y6SNDVk2q
```

- The command:

```
ldapsearch -L -s one -b "c=US" "o=university*" o description
```

Performs a one-level search at the `c=US` level for all organizations whose `organizationName` begins with university. Search results are displayed in the LDIF format. The `organizationName` and `description` attribute values are retrieved and printed to standard output, resulting in output like the following:

```
dn: o=University of Alaska Fairbanks, c=US
o: University of Alaska Fairbanks
description: Preparing Alaska for a brave new tomorrow
description: leaf node only

dn: o=University of Colorado at Boulder, c=US
o: University of Colorado at Boulder
description: No personnel information
description: Institution of education and research

dn: o=University of Colorado at Denver, c=US
o: University of Colorado at Denver
o: UCD
o: CU/Denver
o: CU-Denver
description: Institute for Higher Learning and Research

dn: o=University of Florida, c=US
o: University of Florida
o: UFl
description: Shaper of young minds
...
```

- The command:

```
ldapsearch -h ushost -M -b "c=US" "objectclass=referral"
```

Performs a subtree search for the `c=US` subtree within the server at host `ushost` (TCP port 389) and returns all referral objects. Note the search is limited to the single server. No referrals are followed to other servers to find additional referral objects. The output might look something like this if two referral objects are found:

```
o=IBM,c=US
objectclass=referral
ref=ldap://ibmhost:389/o=IBM,c=US

o=XYZ Company,c=US
objectclass=referral
ref=ldap://XYZhost:390/o=XYZ%20Company,c=US
```

- The command:

```
ldapsearch -b "o=Deltawing, c=AU" -o sn -o -ibm-slapdDN -q 3 -q 2 -T 10
 -v "(|(sn=Harris)(sn=Stephens))" sn
```

Performs a verbose sorted and paged search sorting first by family name, then by distinguished name, with the distinguished name being sorted in reverse (descending) order as specified by the prefixed minus sign. The first page contains 3 entries, the second page contains 2 entries, and the last page contains the final entry, where six entries were found. Each subsequent page is requested 10 seconds after the preceding page is returned. The output might look something like this:

Note: In the following example output, text in this **format** are the returned search entries, while text in this format is the verbose **ldapsearch** utility output.

```
-q option implies -R option. Referrals will not be followed.
ldap_init(MYHOST, 389)
filter pattern: (|(sn=Harris)(sn=Stephens))
returning: sn
sorted search keys: sn -ibm-slapdDN
paged search sizes: 3 2
paged search time: 10
filter is: ((|(sn=Harris)(sn=Stephens)))
cn=Virginia Harris, o=Deltawing, c=AU
sn=Harris

cn=Michael Harris, o=Deltawing, c=AU
sn=Harris

cn=Martin Harris, o=Deltawing, c=AU
sn=Harris
3 matches
3 total paged entries have been returned
6 entries in entire paged results set
The next page will be retrieved in 10 seconds...
cn=Paul Stephens, o=Deltawing, c=AU
sn=Stephens
cn=David Stephens, o=Deltawing, c=AU
sn=Stephens
2 matches
5 total paged entries have been returned
6 entries in entire paged results set
The next page will be retrieved in 10 seconds...

cn=Brian Stephens, o=Deltawing, c=AU
sn=Stephens
1 matches
6 total paged entries have been returned
6 entries in entire paged results set
```

- The command:

```
ldapsearch -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -b "profiletype=user,sysplex=sysplexa" "racfid=G*"
```

Performs a search in the user subtree of the z/VM LDAP support for RACF access for the RACF users whose names begin with `G`. Only the DN of each matching entry is displayed. The z/VM LDAP support for RACF access suffix is assumed to be `sysplex=sysplexa`. The output might look like:

```
racfid=G\#126,profiletype=USER,sysplex=sysplexa
racfid=GDCEBLD,profiletype=USER,sysplex=sysplexa
racfid=GKUPERM,profiletype=USER,sysplex=sysplexa
racfid=GLDSRV,profiletype=USER,sysplex=sysplexa
...
```

To then retrieve the entire entry for one of the matching users, use the command:

```
ldapsearch -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -b "racfid=gldsrv,profiletype=user,sysplex=sysplexa" "objectclass=*"
```

The results might look like:

```
racfid=GLDSRV,profiletype=USER,sysplex=sysplexa
racfid=GLDSRV
racfauthorizationdate=05/15/07
racfowner=RACFID=SUIMGVD,PROFILETYPE=USER,SYSPLEX=SYSPLEXA
racfpasswordinterval=186
racfdefaultgroup=RACFID=AUDIT,PROFILETYPE=GROUP,SYSPLEX=SYSPLEXA
racflogondays=SUNDAY
racflogondays=MONDAY
racflogondays=TUESDAY
racflogondays=WEDNESDAY
racflogondays=THURSDAY
racflogondays=FRIDAY
racflogondays=SATURDAY
racflogontime=ANYTIME
racfconnectgroupname=RACFID=AUDIT,PROFILETYPE=GROUP,SYSPLEX=SYSPLEXA
racfhavepasswordenvelope=NO
racfhavepassphraseenvelope=NO
racfattributes=PASSWORD
objectclass=TOP
objectclass=RACFBASECOMMON
objectclass=RACFUSER
```

The following examples use file input to perform multiple searches with the same search base and scope, but with different filters. Each line in the file is used to replace the first occurrence of `%s` in the filter. The `%s` can be anywhere in the filter, and can be the entire filter.

- Assume file `/tmp/searchFile` has the following contents:

  ```
  Smith
  Jones
  Doe
  ```

  The command:

  ```
  ldapsearch -f /tmp/searchFile -L -s sub -b "o=My Company" "(&(cn=John)(sn=%s*))"
  ```

  replaces the `%s` in the filter value with each line of the input file. This is equivalent to issuing these search commands:

  ```
  ldapsearch -L -s sub -b "o=My Company" "(&(cn=John)(sn=Smith*))"
  ldapsearch -L -s sub -b "o=My Company" "(&(cn=John)(sn=Jones*))"
  ldapsearch -L -s sub -b "o=My Company" "(&(cn=John)(sn=Doe*))"
  ```

- Assume file `/tmp/searchFile` has the following contents:

  ```
  o=university*
  cn=Karen Smith
  ```

  The command:

  ```
  ldapsearch -f /tmp/searchFile -s one -b "c=US" "%s" description
  ```

  replaces the entire filter with each line of the input file. This is equivalent to issuing these search commands:

  ```
  ldapsearch -s one -b "c=US" "o=university*" description
  ldapsearch -s one -b "c=US" "cn=Karen Smith" description
  ```

Searching a server's root DSE

The command:

```
ldapsearch -h ushost -s base -b "" "objectclass=*"
```

provides the root DSE (DSA-specific entry, where a DSA is a directory server) information for a server. This request can be directed to servers supporting LDAP Version 3 protocol to obtain information about support available in the server. See IETF RFC 2251: *Lightweight Directory Access Protocol (v3)* for a description of the information provided by the server. See *z/VM: TCP/IP LDAP Administration Guide* for more information about the root DSE.

The command:

```
ldapsearch -h ushost -s sub -b "" filter
```

searches the directories within the LDAP server for entries that match the filter. This type of search is commonly referred to as a null-based subtree search. See *z/VM: TCP/IP LDAP Administration Guide* for more information about the z/VM LDAP server support for null-based subtree searches.

Note: The scope option (**-s**) must be specified when specifying `-b ""` to search a server's root DSE.

Return Codes

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

Usage Notes

1. The `LDAP_DEBUG` environment variable can be used to set the debug level. For more information about specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see "Enabling Tracing" on page 143.
2. You can specify an LDAP URL for *ldapHost* on the **-h** parameter. See *ldap_init()* for more information.
3. For information about SSL/TLS, see "SSL/TLS information for LDAP utilities" on page 139.

Chapter 7. Monitoring the TCP/IP Network

This chapter describes how to use the NETSTAT, RPCINFO, TRACERTE, and PING commands to obtain information from the network.

- The NETSTAT command displays information about the status of the local host.
- The RPCINFO command displays the servers that are registered and operational with any portmapper on your network.
- The TRACERTE command helps you debug network problems.
- The PING command determines the accessibility of a foreign node.

NETSTAT Displaying Local Host Information

The NETSTAT command displays the network status of the local host. NETSTAT displays information about TCP/IP configuration, connections, network clients, gateways, devices, and the Telnet server. NETSTAT also drops connections and executes commands for users in the TCPIP virtual machine's OBEY list. For more information about the OBEY list, see *z/VM: TCP/IP Planning and Customization*.

This section describes the NETSTAT parameters and contains examples of the responses that are displayed as a result of issuing the NETSTAT command with a specified parameter.

Note: For inverse name resolution, the NETSTAT command uses the ETC HOSTS file rather than the domain name server. If ETC HOSTS does not exist, the HOSTS ADDRINFO file is used.

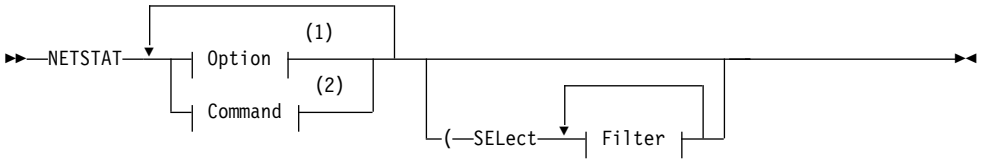
NETSTAT Command Address Interpretation

The NETSTAT command interprets and displays internet addresses symbolically, if possible. If an address cannot be interpreted, it is displayed in dotted decimal notation. Unspecified addresses are displayed as an asterisk (*).

Known port numbers are symbolically displayed. Port numbers that are not known to the network are displayed numerically. A socket is displayed as an internet address, followed by two periods (..) and a port number.

© Copyright IBM Corp. 1987, 2016

NETSTAT Command

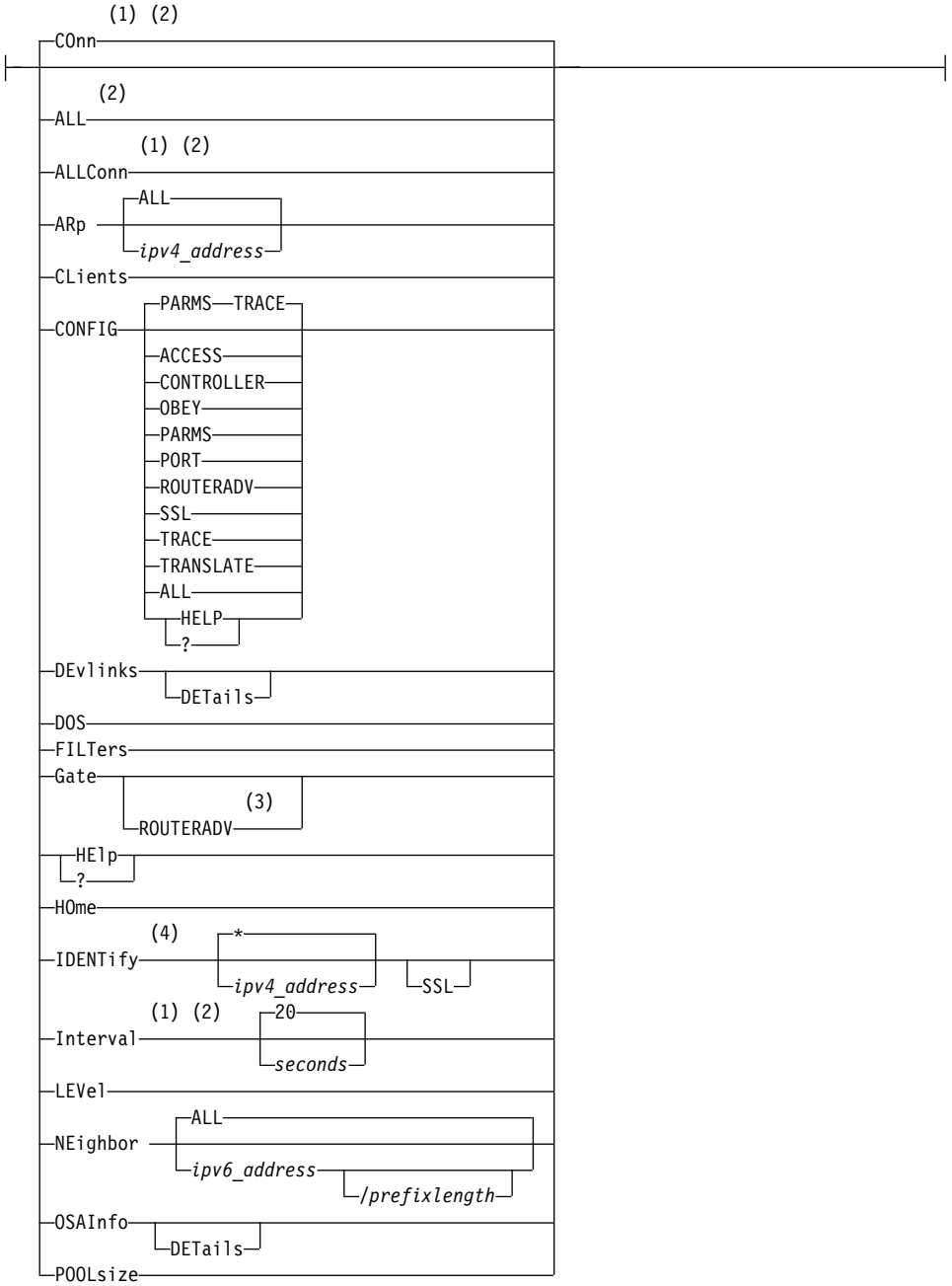


Notes:

1 If multiple options are specified, NETSTAT may not display the output in the sequence as the options were specified on the command line.

2 Only user IDs with TCP/IP OBEY command authorization can use command operands.

Option:



Notes:

1 Only ALLCON, CONN and TCP are valid with INTERVAL.

2 The *userid* filter is valid with ALL, ALLCONN, CONN, and INTERVAL.

3 The *ipv4_address* and *ipv6_address* filters are valid with GATE only.

4 The IDENTIFY option cannot be specified with other options (except TCP).

Option (continued):

Command:

Filter:

Notes:

1 Only ALLCON, CONN and TCP are valid with INTERVAL.

2 The *ldev* filter is valid with TELNET only.

3 All data that follows the CP keyword is used as CP command input.

4 The *ipv4_address* and *ipv6_address* filters are valid with GATE only.

5 The *userid* filter is valid with ALL, ALLCONN, CONN, and INTERVAL.

6 The *devicename* and * filters are valid with OSAINFO only.

Purpose

Use the NETSTAT command to display network status of the local host.

Note: The minimum abbreviation for each parameter is shown in uppercase letters.

Operands

ALL

Displays information about all TCP/IP connections. This option is useful for debugging the TCPIP virtual machine. For more information about maintaining the TCPIP virtual machine, see *z/VM: TCP/IP Planning and Customization*.

Displays the following information for UDP sockets being used for:

- Outgoing Multicast Data
  - the time-to-live value
  - whether datagrams are sent to loopback
  - the IP address of the link on which the datagrams are sent
- Incoming Multicast Data
  - the multicast groups (by IP addresses) for which data is being received.
  - the IP address of the associated link

ALLConn

Specifies that information about connections in either the "closed" or "time-wait" state should be provided, in addition to that for *active* TCP/IP connections (that is, connections that are not in either of these states).

ARp *ipv4_address*

Displays the ARP cache entry for the designated IPv4 address or set of IPv4 addresses. To display entries for multiple IPv4 addresses, specify the last token of the IPv4 address as an asterisk (*).

Example: An *ipv4_address* value of `9.130.48.*` displays ARP cache entries for IPv4 addresses from 9.130.48.0 through 9.130.48.255, whereas `9.*` displays ARP cache entries for network 9. Specifying **ALL** or a single asterisk (*) displays all ARP cache entries.

Note:

1. Entries for network adapters that maintain their own ARP cache are displayed *after* those that are maintained within the TCP/IP server. The time at which TCP/IP last obtained adapter-maintained data precedes these types of entries.
2. The TCP/IP server requests adapter-maintained ARP data upon each NETSTAT ARP invocation. This data is not displayed for an initial NETSTAT ARP command. To display current adapter-maintained ARP data, issue at least two NETSTAT ARP commands, in sequence.

ARp ALL

Displays the ARP cache entry for all IPv4 addresses and also the *arp age* value. An asterisk (*) can be used for this purpose as well.

BLock *ipv4_address*

Ignores incoming traffic from the designated IPv4 address or set of IPv4 addresses. To block traffic from multiple IPv4 addresses, specify the last token of the IPv4 address as an asterisk (*).

Example: An *ipv4_address* value of `9.130.48.*` blocks traffic from IP addresses 9.130.48.0 through 9.130.48.255, whereas `9.*` blocks traffic from network 9.

Note: You should exercise care when selecting IP addresses that are to be blocked. It is possible to block one's own IP address, resulting in unpredictable behavior.

BLock *ipv6_address*
Ignores incoming traffic from the designated IPv6 address. The IPv6 address is specified in full or compressed form.

BLock *ipv6_address/prefixlength*
Ignores incoming traffic from the designated set of IPv6 addresses. To block traffic from multiple IPv6 addresses, specify a prefix by putting a slash (/) immediately after the IPv6 address followed by a *prefixlength*.

Example: Specifying **NETSTAT BLOCK** with an *ipv6_address/prefixlength* value of `1080:5F00:AA:BB:CC::/80` blocks traffic from IP addresses 1080:5F00:AA:BB:CC:0:0:0 through 1080:5F00:AA:BB:CC:FFFF:FFFF:FFFF.

CLients
Displays the following information about each client:

- Authorization, as known by the TCP/IP server; possible values are:

  Autologged
  Client is listed in the AUTOLOG list, so can be autologged by the TCP/IP server.

  Informed
  Client is listed in the TCP/IP INFORM list; it may receive error notifications.

  Monitor
  Client is listed in the TCP/IP OBEY list; it can issue TCP/IP monitor command requests that should be obeyed.

  Probed
  Client supports connection probe notices.

  No-garbage-collect
  Resources in use by this client will not be affected by TCP/IP "garbage collection" activity.
- Notes handled by the client
- Elapsed time since the client was last used
- Elapsed time since the client was last forced (applies only to clients in the AUTOLOG list)
- VMCF error count

CONFIG
Displays configuration information about the TCP/IP server virtual machine. The following operands can be specified in any order following CONFIG, except the ALL operand, which must be specified alone. If no operands are specified, the default is PARMS TRACE.

ACCESS
Displays the users that have access to TCP/IP services. A list of users permitted to use the system, or a list of users restricted from using the system will be displayed. The list displayed is dependent on whether the PermittedUsersOnly option is specified on the AssortedParms statement in the TCP/IP configuration file.

CONTROLLER
Displays the current VSWITCH CONTROLLER settings.

OBEY
Displays the list of users that have access to privileged TCP/IP services.

PARMS
Displays the current AssortedParms and InternalClientParms settings.

AssortedParms and InternalClientParms settings displayed may not match those specified in the TCP/IP configuration file. In some cases, the TCP/IP server enables or disables features based on other environmental factors, and the display reflects the parameters as the TCP/IP server currently views them.

PORT
Displays the list of ports that have been configured via the PORT statement.

ROUTERADV
Displays the router advertisement configuration parameters of your TCP/IP stack. For more information, see the ROUTERADV statement and the ROUTERADVPREFIX statement in *z/VM: TCP/IP Planning and Customization*.

SSL
Displays the user IDs of the SSL servers (if any) associated with the subject TCP/IP virtual machine, and the status of these servers (such as whether TCP/IP currently is using a given SSL server to provide secure connection services). For an SSL server that currently is active, the number of connections managed by that server also are displayed.

The possible states cited for the **Status** field in the command response are:

Active
The server is logged on and has established appropriate connections with the TCP/IP server; dependent upon the current configuration and connection activity, the server may or may not be able to accept new connections.

Eligible
The server is one that has been defined for use, but does not meet the criteria for an active server. It could be that the server currently is not logged on, or the server has not initialized and established the necessary connections with the TCP/IP server. The TCP/IP server will make attempts to employ such a server when a new secure connection request cannot be accommodated by the currently active server(s). A server in this state cannot respond to SSLADMIN administrative commands or accept any connections.

Starting
The server has been started by the TCP/IP server as part of its own initialization, or by a NETSTAT SSL START or SSLADMIN START command. The server currently is progressing through the initialization process. A server in this state is not fully initialized, so cannot respond to SSLADMIN administrative commands or accept any connections.

Stopped
A stopped server is one that has been overtly stopped via an SSLADMIN STOP command. As such, the TCP/IP server will not attempt to employ this server to accommodate a new secure connection request that cannot be handled by the currently active server(s). A server in this state is not logged on, so cannot respond to SSLADMIN administrative commands or accept any connections.

TRACE
Displays the current status of TCP/IP tracing. Information includes the destination of trace data, items currently being traced, as well as the users, devices, and IP addresses that have been configured for selective tracing.

TRANSLATE
Displays information on internet address to MAC address translations that have been configured via the TRANSLATE statement.

ALL
Displays all of the above reports.

HELP
? Displays NETSTAT CONFIG usage information.

COnn
Displays the following information about each *active* TCP/IP connection:
- User ID
- Connection number
- Local socket
- Foreign socket
- Connection state

TCP/IP considers a connection to be *active* if it is not in a "closed" or "time-wait" state.

CONN is the default parameter.

Note: In a Secure Socket Layer (SSL) session, there are two connections — the connection from the remote client to the security server and the connection from the security server to the application server. For each of these connections, the connection number for the partner connection of the secure session is displayed on the next line.

CP *cp_command*
Specifies a CP command to be issued by the TCP/IP server; all data that follows the CP parameter is construed to be part of the CP command.

Example: To close the console of the TCPIP virtual machine and send this output to a specific user ID, use this NETSTAT command:

```
netstat cp spool cons close to userid
```

Up to 32766 bytes of the CP command response are displayed by the NETSTAT command.

Note: CP commands can be used only by privileged TCP/IP users, as identified by the TCP/IP server's OBEY statement. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.

DELarp *ipv4_address*
Deletes the ARP cache entry for the designated IPv4 address or set of IPv4 addresses. To delete entries for multiple IPv4 addresses, specify the last token of the IPv4 address as an asterisk (*).

Example: An *ipv4_address* value of `9.130.48.*` deletes ARP cache entries for IP addresses from 9.130.48.0 through 9.130.48.255, whereas `9.*` deletes ARP cache entries for network 9, and `*` deletes all ARP cache entries.

Note:

1. The DELARP command can be used by privileged TCP/IP users only. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.
2. Adapter-maintained ARP entries cannot be deleted using the NETSTAT DELARP command.

DELNeighbor *ipv6_address*
Deletes the neighbor cache entry for the designated IPv6 address. The *ipv6_address* is specified in full or compressed form.

DELNeighbor *ipv6_address/prefixlength*
Deletes the neighbor cache entry for the designated set of IPv6 addresses. To delete entries for multiple IPv6 addresses, specify an IPv6 address that is immediately followed by a slash (/) and a prefix length. Specifying **ALL** or a single asterisk (*) will delete the entire NEIGHBOR cache.

Example: Specifying **NETSTAT DELNEIGHBOR** with an *ipv6_address/prefixlength* value of `1080:5F00:AA:BB:CC::/80` will delete the neighbor cache entries for all of the IP addresses from 1080:5F00:AA:BB:CC:0:0:0 through 1080:5F00:AA:BB:CC:FFFF:FFFF:FFFF.

DELNeighbor ALL
Deletes all neighbor cache entries. An asterisk (*) can be used for this purpose as well.

DEvlinks
Displays the following information about the devices and links defined for the TCP/IP virtual machine:

- Address
- ARP query capability
- Broadcast capability
- Byte counts
- CPU number
- Device name
- Device type
- HiperSockets port name
- GVRP status
- IPv4 VLAN ID
- IPv6 status
- IPv6 VLAN ID
- Link name
- Link type
- Maximum frame size
- MTU size
- Multicast specific information
- Net number
- Packet forwarding status
- Path MTU discovery status
- Port number
- QDIO port name and router type
- Queue size
- Status
- VLAN ID

The Multicast Specific field is significant only for multicast-capable links. If a link is being used to receive multicast data, then all the multicast groups, and the counts of receivers for each multicast group are displayed.

Special devices and links are described in the output. These are not created as a result of user-specified DEVICE and LINK statements. When a z/VM TCP/IP stack is defined as eligible to be a controller for an OSA-Express device assigned to a virtual switch, a device with type 'VSWITCH-IUCV' shows in the output. For each virtual switch with an active OSA-Express device, a device with type 'VSWITCH-OSD' is described. These devices and links are not explicitly defined in a z/VM TCP/IP configuration file. They are

created as a result of the CP DEFINE VSWITCH command that defines a virtual switch's interface to a network using an OSA-Express device. For a virtual switch with an active BRIDGEPORT device, a device with type 'VSWITCH-HIP' will be displayed. As with 'VSWITCH-OSD', these devices and links are not explicitly defined in a z/VM TCP/IP configuration file. They are created as a result of the BRIDGEPORT RDEV operand on the DEFINE and SET VSWITCH command that defines a virtual switch's bridge interface to a HiperSockets CHPID.

Some fields of the DEVLINKS display are device-dependent. These exceptions are described in the list that follows.

Address
The base address is displayed for all devices.

ARP query capability
Whether or not support for querying ARP address information is shown only for OSD or Hipersockets devices.

Byte counts
Some device drivers do not provide counts. This is not displayed for a device type of VSWITCH-OSD.This is not displayed for a device type of VSWITCH-OSD or VSWTICH-HIP.

GVRP status
Indicates whether GVRP initialization completed successfully. This is displayed only for QDIO Ethernet links that requested GVRP Support, through explicit specification on the LINK statement or by taking the default.

IPv6 status
Reports whether IPv6 support is enabled or disabled for the link. This field is shown only for OSD type devices with the QDIOETHERNET link type and HIPERS type devices with the QDIOIP link type.

IPv4 VLAN ID
IPv6 VLAN ID
A QDIO Ethernet link can be defined to be a member of two Virtual Local Area Networks (VLANs) by configuring one VLAN ID for IPv4 and one for IPv6. This is displayed only for QDIO Ethernet links.

MAC *macaddr*
Displays the MAC address assigned. This is displayed only for QDIO Ethernet links operating in layer 2 mode.

Maximum frame size
The maximum frame size in use. This field is displayed for HiperSockets or VSWITCH-HIP only.

MTU size
The MTU size associated with the link that was either specified as an MTU operand on the LINK statement or was an intelligent default assigned by TCP/IP. This field is shown for all devices except virtual devices.

Net number
This is an integer that identifies the relative adapter number of a network adapter within an LCS device, for which a link is defined. The value is 0 for the first adapter in the LCS, 1 for the second adapter, and so on. This field is significant only for links defined for LCS devices.

Packet forwarding status
Reports whether IP forwarding is enabled or disabled for the link.

Path MTU discovery status
Indicates whether path MTU discovery is enabled or disabled. It is displayed for all non-VIPA devices.

Port number
This is an integer that identifies the OSA-Express hardware port number within a QDIO OSD device. This field is significant only for links defined for OSD devices.

Status Some device drivers do not provide device-specific status. For these devices, possible status values are:
Active The device is started.
Inactive
The device is not started.

Transport Type
IP
ETHERNET
Indicates the transport type specified. IP transport operates at a layer 3 level of the OSI model while ETHERNET operates at a layer 2 level. This is displayed only for QDIO Ethernet, VSWITCH-OSD or VSWITCH-HIP links.

VLAN ID
A QDIO Ethernet or QDIO IP link can be defined to be a member of a single Virtual Local Area Network (VLAN). This is displayed only for QDIO Ethernet and QDIO IP links.

DETails
Displays detailed packet statistics in addition to normal NETSTAT DEVLINKS output.

DOS
Displays information about Denial-of-Service attacks. For each attack detected, it displays the following information:

- The name of the attack
- Up to seven IP addresses from which the attack was launched. (For Fraggle, Smurf-IC, Smurf-OB, and Smurf-RP, the IP address is spoofed to be the address of the victim of the attack. Other denial-of-service attacks may also spoof the source IP address to be the address to be something other than the address of where the attack originated.)
- An IP address of * to represent any additional IP addresses that are not already displayed.
- The number of attacks detected per IP address.
- The elapsed time since the first attack was detected.
- The duration of the attacks.
- The maximum number of half-open connections that are allowed. See the "PENDINGCONNECTIONLIMIT" configuration statement in *z/VM: TCP/IP Planning and Customization* for more information.
- The maximum number of persist connections that are allowed. See the "PERSISTCONNECTIONLIMIT" configuration statement in *z/VM: TCP/IP Planning and Customization* for more information.
- The maximum number of connections that a foreign IP address is allowed to have open at the same time. See the "FOREIGNIPCONLIMIT" configuration statement in *z/VM: TCP/IP Planning and Customization* for more information.

DRop *conn_num*
Drops the TCP/IP connection specified by *conn_num*. You determine the connection number to be dropped from the CONN column of the NETSTAT CONN or NETSTAT TELNET display. If you drop the "passive open" connection for a server, that server will immediately reissue an "open" request.

Note: The DROP command can be used only by privileged TCP/IP users. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.

FILTers
Displays information about blocked IP addresses. For each blocked address or set of addresses, it provides the following information:

- The IP address
- The number of packets discarded
- How long the block has been in effect
- How long ago the last packet from the IP address was discarded

Gate
Displays information about gateways (dynamic routes) known by the TCP/IP server.

Note: If the TCP/IP server knows IPv6 routes from router advertisements which indicated route preference values, then the TCP/IP server only makes use of a subset of these IPv6 routes; that is, all reachable IPv6 routes with the best preference value advertised. NETSTAT GATE displays only this subset of routes. For example, the subset used and displayed might get dropped and replaced by a subset of less favorable IPv6 routes if all other IPv6 routes with a more favorable route preference value have been deleted or have become unreachable. The TCP/IP server supports only IPv6 route preference values for default routes. (This is support for a type B host as defined in RFC 4191).

The following information is displayed for each gateway:

- Address of the network
- The prefix length (for IPv6)
- First hop address
- Flags

  A The route was created dynamically by a Router Advertisement (IPv6 only).

  C The route was created dynamically by a redirect.

  G The route is to a gateway

  H The route is to a host rather than to a network.

  M The route was modified dynamically by a redirect.

  P The route has been created by path MTU discovery.

  Q The packet size used by the route has been modified by path MTU discovery.

  S The route is a static route. Only routes defined using the GATEWAY configuration statement are flagged as static.

  T The route was generated automatically based on the TCP/IP stack configuration and is replaced if a matching route (either static or dynamic) is added.

U The route is up.

- Link name used by the first hop
- Packet size used by the first hop

  Note: If the MTU option is specified on the LINK statement, the MTU shown by NETSTAT GATE matches the MTU shown by NETSTAT DEVLINKS and the ifMtu fields in the TCP/IP APPLMON Monitor Records. Otherwise, these values might differ.
- Subnet mask and subnet value (IPv4 only)
- Route metric:
  - If the route is an OSPF inter-area or intra-area route, this is the OSPF cost of the route.
  - If the route is an OSPF External type 1, this is the OSPF cost to the AS Boundary Router or Forwarding address that is used to reach the destination, plus the external cost.
  - If the route is an OSPF External type 2, this is the external cost.
  - If the route is RIP, this is the RIP metric.
  - If this is a router advertisement route, this is the route preference (IPv6 only):

    1 If the router advertisement indicated a preference of high.

    2 If the router advertisement indicated a preference of medium.

    3 If the router advertisement indicated a preference of low.
  - If the route is static, this value is:

    0 if the route is direct (no first hop).

    1 if the route is indirect.

ROUTERADV

Displays information about all IPv6 gateways (dynamic routes) known from Router Advertisements by the TCP/IP server. The information displayed for each IPv6 gateway is similar to the respective output of the NETSTAT GATE command, except for Route Metric, which is not displayed. Instead the following information is shown:

- Route preference advertised by the first hop:

  Low The first-hop router has advertised a preference of low.

  Medium
  The first-hop router has advertised a preference of medium

  High The first-hop router has advertised a preference of high.
- Route reachability (default routers only):

  Yes The first-hop router is assumed to be reachable.

  No The first-hop router is assumed to be unreachable.

  Probing
  The first-hop router is assumed to be unreachable and is being probed.

HElp
?

Displays brief help information about the NETSTAT command and its operands and parameters.

HOme

Displays the HOME list known by the TCP/IP server; an internet address, link name, and subnet mask column are displayed for each entry of the HOME list. When the IP address is a virtual switch management address, the VSWITCH name is also displayed. For more information about the HOME list (and the HOME statement), see *z/VM: TCP/IP Planning and Customization*.

PI

IDENTify *ipv4_address*
IDENTify *ipv4_address* **SSL**

Produces information identifying the connections associated with a given IPv4 address or set of IPv4 addresses. To select a set of IPv4 addresses, specify the last token of the IPv4 address as an asterisk (*).

Example: To produce information about all IPv4 addresses that begin with 9.148.134, use the operand `9.148.134.*`.

The output from IDENTIFY is intended for programmable use and as such is delimited with blanks rather than formatted in columns.

The following information is given for each connection:

- Local IP address
- Local port
- Foreign IP address
- Foreign port
- Client name
- Logical device status (for Telnet connections only; not included when the **SSL** operand is also used.)

To receive information on secure connections, include the **SSL** operand. In addition to the previously described information (up to, and including the client name), the following information (in order) is added for connections that can be categorized as a secure session:

- Connection type identifier; **E** represents an explicit (TLS-negotiated) connection; **I** represents an implicit (non-negotiated) connection.
- SSL Server ID
- Certificate Label. For a connection associated with a z/VM client, the string **<None>** appears in this field.
- Cipher Details
  - Class (Null, SSLV2, SSLV3, TLS)
  - Hash (Null, MD5, SHA1, SHA2)
  - Symmetric Algorithm (Null, RC2, RC4, DES, DES3, AES)
  - PK Algorithm (Null, RSA, DH_DSS, DH_RSA, DHE_DSS, DHE_RSA)
- Keyring (if it exists). When keyring information is present, a semicolon (;) delimits this from previously listed values.

Note:

1. Because secure session information is established and maintained differently for explicit (**E**) and implicit (**I**) connections, two entries are present in the command output for an implicitly secured session:
   - one entry that identifies the SSL server as a participant, for which security details are included, and

- one entry that identifies the local client participant of the secure session (for which security details are not repeated)

For a secure session established by explicit means, only one entry exists, which identifies both of the participants of the session, as well as its security details.

2. The above-listed security details are not included for subordinate connections, which jointly comprise a secure session — that is, the remote host-to-SSL server and SSL server-to-local application connections.
3. If an explicit secure connection is changed to one that is not secure (as can be done with the FTP CCC command), that connection still is identified as an explicit (**E**) connection, but with additional security details.

PI end

Interval *seconds*

Initiates a full screen display of TCP/IP connections. The screen is updated every *seconds* seconds; the default is 20 seconds. Information may be sorted by idle time (the default), foreign socket, user ID, bytes out, bytes in, or by (connection) state.

The following information is given for each connection:

- User ID
- Bytes sent on the connection
- Bytes received on the connection
- Local port
- Connection State
- Idle time (*hh*:*mm*:*ss*)
- Foreign socket

The number of TCBs in use is displayed at the bottom of the screen.

There are two sets of PF keys for the Interval display screen. The initial set of PF keys are defined as follows:

PF 1 Usr
: Sort by user ID.

PF 2 Sock
: Sort by foreign socket.

PF 3 Quit
: Exit.

PF 4 BOut
: Sort by bytes out (bytes sent on a connection).

PF 5 BIn
: Sort by bytes in (bytes received on a connection).

PF 6 PFSet2
: Show PF key set number 2.

PF 7 Up
: Scroll up (backward) — when more than one screen of information is available for display.

PF 8 Dwn
: Scroll down (forward) — when more than one screen of information is available for display.

PF 9 T/B
: Scroll to top / bottom of data.

PF 10 Rgtleft
: Shift screen data to the right or left (back to the original position).

PF 11 Ip@
Locate function; the line at which the cursor is positioned becomes the first line of displayed information.

PF 12 Rfsh
Refresh connection information.

If you press PF 6 while the first PF key set is displayed, you see the second set of PF keys:

PF 1 St
Sort by connection state.

PF 2 Save
Save data in a file (NETSTAT DATA) and exit.

PF 3 Quit
Exit.

PF 4 Unused
Not set.

PF 5 Unused
Not set.

PF 6 PFSet1
Show PF key set number 1.

PF 7 Up
Scroll up (backward) — when more than one screen of information is available for display.

PF 8 Dwn
Scroll down (forward) — when more than one screen of information is available for display.

PF 9 T/B
Scroll to top / bottom of data.

PF 10 Rgtleft
Shift screen data to the right or left (back to the original position).

PF 11 Unused
Not set.

PF 12 Rfsh
Refresh connection information.

Note: The Enter key performs the same function as PF 3 (Quit).

LEVel
Displays the processor type, z/VM system level, and TCP/IP system level and identifies the stack MODULE (filename, filetype, and filemode) being executed by the TCP/IP server the date when the MODULE was generated and where in storage it is loaded.

NEighbor *ipv6_address*
Displays the neighbor cache entry for the designated IPv6 address. The *ipv6_address* is specified in full or compressed form.

See Note on the NEIGHBOR option.

NEighbor *ipv6_address/prefixlength*
Displays the neighbor cache entry for the designated set of IPv6 addresses. To display entries for multiple IPv6 addresses, specify an IPv6 address that is immediately followed by a slash (/) and a prefix length.

Example: Specifying **NETSTAT NEIGHBOR** with an *ipv6_address/prefixlength* value of `1080:5F00:AA:BB:CC::/80` will display the neighbor cache entries for all of the IP addresses from 1080:5F00:AA:BB:CC:0:0:0 through 1080:5F00:AA:BB:CC:FFFF:FFFF:FFFF.

See Note on the NEIGHBOR option.

NEighbor ALL
Displays the neighbor cache entry for all IPv6 addresses. An asterisk (*) can be used for this purpose as well.

Note on the NEIGHBOR option: The TCP/IP server does not maintain neighbor data for a HiperSockets link. Instead, it requests information from the adapter and creates a NETSTAT NEIGHBOR command display. To display current adapter-maintained neighbor data, issue at least two NETSTAT NEIGHBOR commands in sequence. The TCP/IP server requests adapter-maintained neighbor data upon each NETSTAT NEIGHBOR invocation. This data is not displayed for an initial NETSTAT NEIGHBOR command.

OBEY *statement*
Processes one or more TCP/IP server configuration statements or VSWITCH controller configuration statements as if they had been entered in a file and used as the object of an OBEYFILE command. All data that follows the OBEY parameter is processed as part of the statement string. The OBEY command is subject to the considerations described in *z/VM: TCP/IP Planning and Customization* for use of the OBEYFILE command. In addition, the length of the OBEY command operand is limited to roughly 240 characters, so not all configuration changes can be made using this command.

Whenever you add or change a configuration statement using the OBEY command, be aware that the existing statement is replaced. Therefore, you must supply the entire statement, not just the new or changed portions.

When there is a problem with OBEY, file *StackID* TCPERROR is sent to your reader containing a description of the error and the following error message is displayed:

```
Configuration error. Details are in StackID TCPERROR.
```

Note:

1. The OBEY command can be used only by privileged TCP/IP users, as identified by the TCP/IP server's OBEY statement. For more information about this facility, see *z/VM: TCP/IP Planning and Customization*.
2. Because the OBEY commands are processed asynchronously, you must wait for completion of a command before invoking any subsequent command that depends on a prior command. This is especially important when invoking commands to limit or turn on tracing.

OSAInfo
Displays basic information such as IP and MAC addresses from the OSA Address Table (OAT) for the devices defined for the TCP/IP virtual machine on OSA-Express cards that support the OSAINFO function. The SELECT filter can be used to chose information for a specific device. If no filter is used, the content of the OAT for all connected devices will be displayed.

DETails
Displays all content of the OAT available.

Note:

1. The OSAINFO command can be used by privileged TCP/IP users only. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.

POOLsize
Displays information about free pool control block and data buffer pools. The following information is displayed for each free pool element:
- Name of the free pool element.
- Current maximum number of elements available for use.
- Number of elements available for use.
- "Low water mark" for this element pool; this is the fewest number of elements that have been available since TCP/IP was started.
- Permit size calculated for this element. If the number of elements for a pool drops below the permit size, TCP/IP considers the pool to be running low.

For more information about the free pool, see *z/VM: TCP/IP Planning and Customization*.

RESETDOS
Resets the data displayed for the Denial-of-Service (DOS) attacks.

Note:

1. The RESETDOS command can be used only by privileged TCP/IP users. Such users are identified with the TCP/IP OBEY statement. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.

RESETPool
Resets the "informed" message flags for all *free pool* element pools. This allows pool-related notification messages and mail to again be sent.

When the number of elements for a particular pool drops below its permit size, the TCP/IP server sends a message and mail to all users listed in the INFORM list, and then sets an "informed" flag for that pool. This flag blocks further notifications for the pool, even its number of elements rises above, and then again drops below, the permit size.

Note: The RESETPOOL command can be used only by privileged TCP/IP users. Such users are identified with the TCP/IP OBEY statement. For more information about listing user IDs with the OBEY statement, see *z/VM: TCP/IP Planning and Customization*.

SELect *filter*
Specifies a character string that is used to limit response information to entries associated with one of the following:
- Client or server user ID (*userid*)
- IPv4 address (*ipv4_address*)
- IPv6 address (*ipv6_address*)
- IPv6 address with prefix (*ipv6_address*/*prefixlength*)
- Logical device number (*ldev*)
- Specifies a character string that is used to limit response information to entries associated with one of the following:

  ```
  '*' or 'Device Names'
  ```

The value specified for *filter* can be a complete string, or a partial string terminated by an asterisk (*), which selects information about multiple entries that all begin with *filter*.

Example: To select information that corresponds to only the "default" gateway route known by TCP/IP, specify the *filter* value `default` for a NETSTAT GATE command, as follows:

```
netstat gate (select default
```

You can specify up to six unique *filter* values, each of which can be up to 40 characters long. If specified, the SELECT operand and its *filter* value(s) must be the last parameters of the NETSTAT command.

SOCKets
Displays information about each client using the socket interface.

Each set of one or more sockets is preceded in the response by a socket descriptor. The descriptor contains the following information:

Name: The virtual machine User id of the socket set owner.

Subtask:
Identifies the client application

Path id:
The IUCV path identifier

Pending call:
The name of the active socket function, if any.

The following information is displayed for each socket:

Sock Is the socket number.

Type Indicates the socket type, such as stream (TCP) sockets, datagram (UDP) sockets, raw sockets, or the special SNMP DPI socket type used only by SNMP agents.

Bound to
Shows the address and port to which the socket is bound. Unbound TCP and UDP sockets are not displayed by the NETSTAT CONN or NETSTAT INTERVAL commands.

Connected to
Shows the address and port to which the socket is connected.

State Displays the TCP connection state for TCP sockets. For raw sockets, the IP protocol number is displayed as well. If the *State* field is blank, the *Conn* field is also blank.

Flgs Connection flag (displayed for TCP sockets only); possible values are:
- **A** Indicates a connection on the almost accept queue.
- **C** Indicates a connection on the accept queue.
- **L** Indicates a listening socket.

Conn Displays the internal TCP control block (TCB) number used by the TCP/IP server for this connection. *Conn* applies only to TCP sockets. If this field is blank, the flag for this connection will be an **L**; this indicates this is a listening socket for which the accept queue is full, or for which the TCP/IP server is temporarily unable to allocate resources to put a TCB in a "Listen" state. Attempts to connect to the port (displayed in the *Bound to* field) are ignored; this allows TCP to retry the connection.

SSL START *userid*
Initiates the controlled initialization of an SSL server, or all such servers that are members of a defined server pool.

When multiple NETSTAT SSL START commands are used (or *userid* is specified as ALL), the start up of all servers — other than the first identified server — is delayed, to avoid contention for common resources that are shared among the various pool servers.

TCp *server*
Identifies the TCP/IP stack server or the VSWITCH controller server for which status information is to be displayed, or to which commands are to be directed.

TELnet
Displays the status of the internal Telnet server.

UNBLock *ipv4_address*
Allows incoming traffic from the designated IPv4 address or set of IPv4 addresses by removing a previously defined filter. The specification must match the original definition.

UNBLock *ipv6_address*
Allows incoming traffic from the designated IPv6 address. The specification must match the original definition.

UNBLock *ipv6_address/prefixlength*
Allows incoming traffic from the designated set of IP addresses by removing a previously defined filter. The specification must match the original definition.

Up Displays the date and time that TCP/IP was started.

Return codes

The following is a list of the return codes generated by NETSTAT:

Code	Description
0	Normal completion; no errors encountered.
1	Syntax error.
4	NETSTAT command error.
8	Execution error–runtime error.
12	TCP/IP error.
20	User is not authorized to issue command.
100	Command line is too long.
1*nnnn*	*nnnn* is a CP return code from NETSTAT CP.

Examples

This section shows sample responses for various NETSTAT commands, issued with a specific operand or set of operands.

ALL

The following is a sample of the information that is displayed, in this case for two clients after entering NETSTAT ALL.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Client: FTPSERVE                      Last Touched:    2:07:08
Local Socket: *..FTP-C                Foreign Socket: *..*
     BackoffCount: 0
     ClientRcvNxt: 0
     ClientSndNxt: 713933277
     CongestionWindow: 65535
     Local connection name: 1002
     Sender frustration level: Contented
     Incoming window number: 0
     Initial receive sequence number: 0
     Initial send sequence number: 0
     Maximum segment size: 536
     Outgoing window number: 0
     Precedence: Routine
     RcvNxt: 0
     Round-trip information:
        Smooth trip time:   0.000
        Smooth trip variance:   1.500

                                                          More...   BTP311S6
```

```
     SlowStartThreshold: 65535
     SndNxt: 713933276
     SndUna: 713933276
     SndWl1: 0
     SndWl2: 0
     SndWnd: 0
     MaxSndWnd: 0
     State: Listen
     No pending TCP-receive

Ready;
```

ALLCONN

The following is a sample of the information that is displayed after entering NETSTAT ALLCONN.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Active IPv4 Transmission Blocks:

User Id  Conn    Local Socket            Foreign Socket          State
---- --  ----    ----- ------            ------- ------          -----
FTPSRV70 1002    *..FTP-C                *..*                    Listen
INTCLIEN 1000    *..TELNET               *..*                    Listen
PORTMP70 UDP     *..PMAP                 *..*                     UDP
PORTMP70 1003    *..PMAP                 *..*                    Listen
VMNFS70  UDP     *..2049                 *..*                     UDP
VMNFS70  1004    *..2049                 *..*                    Listen
MOYWLNX1 UDP     Loopback..8080          *..*                     UDP
MOYW     1005    Loopback..3090          *..*                    Closed

Active IPv6 Transmission Blocks:

User Id  Conn     State
---- --  ----     -----
MOYWLNX1 1001     Listen
  Local Socket:   *..9999
  Foreign Socket: *..*
MOYWLNX1 UDP       UDP
  Local Socket:   ::1..6090
  Foreign Socket: *..*
MOYW     UDP       UDP
  Local Socket:   *..4040
  Foreign Socket: *..*
MOYW     1006     Closed
  Local Socket:   ::1..9370
  Foreign Socket: *..*

Ready;
```

ARP

The following is a sample of the information that is displayed after entering NETSTAT ARP 9.117.*

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

ARP Age: 5

Querying ARP cache for address 9.117.*
Link TR1             : IBMTR: 08-00-5A-8B-32-2E IP: 9.117.32.15
   Route info: 8220
Link TR1             : IBMTR: 00-04-AC-20-52-1C IP: 9.117.32.29
   Route info: 0000
Link TR1             : IBMTR: 40-00-00-57-FD-BC IP: 9.117.32.249
   Route info: 0592
Link ETH1            : ETHERNET: 42-60-8C-2C-E2-22 IP: 9.117.176.4

Adapter-maintained data as of: 05/25/05 10:00:59

LINK OSDQDIO2        : QDIOETHERNET: 10-00-5A-99-8C-6B IP: 9.117.176.1
LINK OSDQDIO2        : QDIOETHERNET: 00-04-AC-7C-82-F5  IP: 9.117.176.245
LINK OSDQDIO2        : QDIOETHERNET: 00-04-AC-7C-82-F5  IP: 9.117.248.157
Ready;
```

BLOCK

The following is a sample of the information that is displayed after entering NETSTAT BLOCK 9.130.48.*

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Function performed
Ready;
```

CLIENTS

The following are examples of the information that is displayed for a client after entering NETSTAT CLIENTS.

For a client with notes handled:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Current clients:
Client: FTPSERVE                      Authorization: Autologged
Notes Handled: Buffer space available, Connection state changed, Data delivered,
Urgent pending, Other external interrupt received, Timer expired,
FSend  response
FReceive erro, IUCV interrupt
Last Touched:   2:20:07               Last Forced: 2:36:59
Vmcf error count: 0
Ready;
```

For a client with no notes handled:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Current clients:
Client: OPERATOR                      Authorization: Monitor, Informed
Notes Handled: none
Last Touched:   2:17:43               Last Forced: 2:34:23
Vmcf error count: 0
Ready;
```

CONFIG Statement

The following are examples of the information that is displayed after entering NETSTAT CONFIG ACCESS or NETSTAT CONFIG CONTROLLER.

ACCESS: The following is a sample of the information that is displayed after entering NETSTAT CONFIG ACCESS:

```
netstat config access
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Users restricted from using TCP/IP services:

BADGUY   HACKER    SPAMKING

Ready;
```

CONTROLLER: The following is a sample of the information that is displayed after entering NETSTAT CONFIG CONTROLLER:

```
netstat config controller
VM TCP/IP Netstat level 640       TCP/IP Server Name: DTCVSW1

Vswitch Controller Status: Available
                           Device Address Range: * *
                           Failover Enabled
```

OBEY: The following is a sample of the information that is displayed after entering NETSTAT CONFIG OBEY:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Privileged TCP/IP Users:

TCPMNT06 OPERATOR  MAINT     TCPMAINT  MPROUT06

Ready;
```

PARMS: The following is a sample of the information that is displayed after entering NETSTAT CONFIG PARMS:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIPB6

Assorted Parameters
  CP Dump:                    No        VM Dump:                    No
  Ignore Redirects:           Yes       IPv6 Ignore Redirects:      No
  ACB Cushion:                No        Forwarding Enabled:         No
  Warn on Level Mismatch:     Yes       RFC1323 Support             Yes
  UDP Queue Limit:            Yes       IPv4 PMTU Default Enabled:  No
  Permitted Users Only:       No        Proxy ARP:                  No
  Restrict Low Ports:         Yes       Secure Local:               No
  Source VIPA:                No        IPv6 Source VIPA:           No
  Check Consistency:          No

Internal Client Settings
  Asynchronous Input:   No              CCS Terminal Name:        TCPIP
  Connection Exit:      <none>          EOJ Time Out:             120
  Go Aheads Disabled:   No              Ignore EAU Data:          No
  Inactivity Timeout:   0               LDev Range:               0000 - 0FFF
  Scan Interval:        60              Timemark Timeout:         0
  TN3270E Enabled:      Yes             Use SC Exit for TN3270E:  No
  TN3270E Exit:         <none>          Transform:                No
  Secure Connection:    ALLOWED         TLS Label:                ZVMCERT
  Client Cert Check:    None
  Port(s):              23      623

Ready;
```

PORT: The following is a sample of the information that is displayed after entering NETSTAT CONFIG PORT:

```
netstat config port
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Configured Port Information:

Port: 5000          Protocol: TCP
IP Address: *
Monitor: Yes
User ID: MIGUELD    Certificate: MDCERT

Port: 520           Protocol: UDP
IP Address: *
Monitor: No
User ID: MPROUT06   Certificate: <none>

Port: TELNET        Protocol: TCP
IP Address: *
Monitor: Yes
User ID: IntClien   Certificate: <none>

Port: 80            Protocol: TCP
IP Address: 24C:4562::2222:F009
Monitor: No
User ID: REDHAT1    Certificate: <none>

Port: 2000 - 2050   Protocol: TCP
IP Address: *
Monitor: Yes
User ID: Reserved   Certificate: <none>

Port: 8080          Protocol: UDP
IP Address: 9.60.59.6
Monitor: No
User ID: TCPIP      Certificate: <none>

Ready;
```

In the preceding sample NETSTAT CONFIG PORT output, the following are definitions for the related fields:

IP Address
IP address which will be used for the port(s). Asterisk (*) means that any IP address may bind to the port(s).

Monitor
This value relates to the NOAUTOLOG parameter of the PORT statement. Value is NO if NOAUTOLOG was specified.

User ID
The user for which this (these) port(s) are reserved. “Reserved” means that no user can listen on the given port(s). IntClien means the port(s) is/are reserved for the TELNET client. Asterisk (*) means any user may listen on the port(s).

ROUTERADV: The following is a sample of the information that is displayed after entering NETSTAT CONFIG ROUTERADV:

```
netstat config routeradv
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

IPv6 Router Advertisement Information:

Link TOVSL2SWA6
  Send Router Advertisements:  On
  Max Multicast Adv Interval:  600
  Min Multicast Adv Interval:  200
  Stateful Autoconfig Address: Off
  Stateful Autoconfig Other:   Off
  Route Preference Value:      Medium
  MTU Size:                    0
  Reachable Time:              0
  Message Retransmission Time: 0
  Current Hop Limit:           64
  Router Lifetime:             3
  Include source link address: On
  Prefix             5C6:C2C1::/64
   On-Link Determination:      On
   Auto Address Determination: On
   Valid Lifetime:             2592000
   Preferred Lifetime:         604800
  Prefix             5C7:C2C1::/64
   On-Link Determination:      On
   Auto Address Determination: On
   Valid Lifetime:             2592000
   Preferred Lifetime:         604800

  Ready;
```

If z/VM is configured to send router advertisement messages (using the ROUTERADV configuration option, the configuration for all ROUTERADV options and for all ROUTERADVPREFIX options is displayed. If z/VM is not configured to send router advertisement messages, the following output will occur:

```
netstat config routeradv
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

IPv6 Router Advertisement Information: None

Ready;
```

SSL: The following is a sample of the information that is displayed after entering NETSTAT CONFIG SSL:

```
netstat config ssl
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP01

SSL Server User ID: SSL00001   Status: Active   Connections: 600
SSL Server User ID: SSL00002   Status: Active   Connections: 600
SSL Server User ID: SSL00003   Status: Active   Connections: 300
SSL Server User ID: SSL00004   Status: Eligible
SSL Server User ID: SSL00005   Status: Stopped
Maximum Session System Limit: 3000
Maximum Session Server Limit:  600
SSL Session High-Water Mark   1500

Ready;
```

TRACE: The following is a sample of the information that is displayed after entering NETSTAT CONFIG TRACE:

```
netstat config trace
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Trace Destination: CONSOLE

Tracing enabled for:
  "ARP, *CCS CP System Service, TCP congestion control,
  Consistency checker, CTC handler, Device driver,
  External interrupt handler, ICMP, Internal Telnet server,
  Internal Telnet timeout handler, Ioctl for routing,
  I/O interrupt handler, IP-down, IP-request, IP-up, IUCV-API-greeter,
  IUCV handler, Monitor, Notify, Common Packet handler, Parse-Tcp,
  PCCA handler, PCCA3 handler, Ping process, RAWIP-request, RAWIP-up,
  Retransmit-datagram, Roundtrip, Shutdown, SNMP DPI sub-agent,
  Sock-request, Status-in, Status-out, TCP-down, TCP-request, TCP-up,
  To-CETI, CTC common routine, To-ELANS, To-ILANS, To-IUCV,
  To-IUCV signon, PCCA common routine, PCCA3 common routine,
  UDP-request, UDP-up, IGMP, Multicast, Dynamic Routing, OSD handler,
  OSD common routine, FPSM process, QDIO process, SSL,
  Denial of Service, IUCV interrupt director, Trace Table, Security"

Detailed Tracing enabled for:
  <none>

Selective Tracing enabled for the following users/devices/IP addresses:
  TCPMAINT
Ready;
```

HELP: The following is a sample of the information that is displayed after entering NETSTAT CONFIG HELP:

```
netstat config help
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Usage: NETSTAT CONFIG options

Options:
ACCESS    - Query users with access to TCP/IP services
CONTROLLER - Query the Controller status of the virtual switch
OBEY      - Query users with access to privileged TCP/IP operations
PARMS     - Query current AssortedParms and InternalClientParms
PORT      - Query reserved and secure port information
ROUTERADV - Query current ROUTERADV settings
SSL       - Query SSL server user ID information
TRACE     - Query currently active TCP/IP tracing
TRANSLATE - Query IP address to MAC address translations
ALL       - Displays all of the above information
HELP or ? - Displays this help text
Note: If no options are specified, the default is PARMS TRACE

Ready;
```

CONN

The following is a sample of the information that is displayed after entering NETSTAT CONN.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Active IPv4 Transmission Blocks:

User Id  Conn    Local Socket            Foreign Socket          State
---- --  ----    ----- ------            ------- ------          -----
FTPSRV70 1002    *..FTP-C                *..*                    Listen
INTCLIEN 1000    *..TELNET               *..*                    Listen
PORTMP70 UDP     *..PMAP                 *..*                     UDP
PORTMP70 1003    *..PMAP                 *..*                    Listen
VMNFS70  UDP     *..2049                 *..*                     UDP
VMNFS70  1004    *..2049                 *..*                    Listen
MOYWLNX1 UDP     Loopback..8080          *..*                     UDP

Active IPv6 Transmission Blocks:

User Id  Conn     State
---- --  ----     -----
MOYWLNX1 1001     Listen
  Local Socket:   *..9999
  Foreign Socket: *..*
MOYWLNX1 UDP       UDP
  Local Socket:   ::1..6090
  Foreign Socket: *..*
MOYW     UDP       UDP
  Local Socket:   *..4040
  Foreign Socket: *..*

Ready;
```

CP

The following is a sample of the information that is displayed after entering NETSTAT CP QUERY TIME.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

CP command output is:
TIME IS 17:27:58 EST WEDNESDAY 07/13/05
CONNECT= 00:23:25 VIRTCPU= 000:03.62 TOTCPU= 000:05.79

CP return code= 0
Ready;
```

Note: You must be a privileged user to use the CP command.

DELARP

The following is a sample of the information that is displayed after entering NETSTAT DELARP 9.130.3.48.

Note: You must be a privileged user to use the DELARP command.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

1 ARP cache entries deleted for 9.130.3.48
Ready;
```

DELNEIGHBOR

The following is an sample of the information that is displayed after entering NETSTAT DELNEIGHBOR FE80::209:5700:100:21:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

1 neighbor cache entries deleted for FE80::209:5700:100:21
Ready;
```

DEVLINKS

The following is a sample of the information that is displayed after entering NETSTAT DEVLINKS.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

 Device TOTCP2               Type: CTC          Status: Ready
   Queue size: 0    CPU: 0   Address: 03F8
     Link VCTC2              Type: CTC          Net number: 0
       Speed: 4500000
       BytesIn: 3356914      BytesOut: 3415727
       Forwarding: Enabled   MTU: 9216
       IPv4 Path MTU Discovery: Disabled
       Broadcast Capability: Yes
       Multicast Capability: Yes
       Multicast Group  Members
       ---------------  -------
       224.67.113.10       3
       225.36.12.9         5

 Device LCS1                 Type: LCS          Status: Ready
   Queue Size: 0    CPU: 0   Address: 09E0
     Link ETH1               Type: ETHERNET     Net number: 0
       Speed: 10000000
       BytesIn: 13965        BytesOut: 38904
       Forwarding: Enabled   MTU: 1500
       IPv4 Path MTU Discovery: Disabled
       Broadcast Capability: Yes
       Multicast Capability: Yes
       Multicast Group  Members
       ---------------  -------
       224.67.113.10       3
       225.36.12.9         5

 Ready;
```

In the preceding example, the first device indicated is `TOTCP2`, which is a device of type `CTC` (a Channel-to-Channel device) that has a base virtual address of `03F8` and whose device driver runs on CPU 0. There is one link defined for this device, named `VCTC2`. This link has LAN broadcast and multicast capabilities.

The second device indicated is `LCS1`, which is a device of type `LCS` (a LAN Channel Station device) that has a base virtual address of `09E0` and whose device driver runs on CPU 0. There are two links defined for this device — `ETH1`, an Ethernet link. The `Net number` (or, *relative adapter* number) for each device is 0; this indicates that this is the first link for this device. The ETH1 link has LAN broadcast and multicast capabilities. There are 3 members in the multicast group 224.67.113.10 and 5 members in the multicast group 225.36.12.9.

The status of these devices is “Ready”, which indicates they are operational. Also, the `Queue Size` of zero for each indicates that no envelopes are queued for output.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Device DEVHS0                  Type: HIPERS         Status: Ready
  Queue size: 0     CPU: 0     Address: 0520        Port name: HSA0PORT
  IPv4 Router Type: NonRouter  Arp Query Support: Yes
    Link LINKHS0               Type: QDIOIP         Net number: 0
      Speed: 100000000
      BytesIn: 620680          BytesOut: 207120
      Forwarding: Enabled      MTU: 8192
      Maximum Frame Size  : 16384
      IPv4 Path MTU Discovery: Disabled
      Broadcast Capability: Yes
      Multicast Capability: Yes
      Multicast Group                        Members
      ---------------                        -------
      224.0.0.1                                 1

Device DEVOSD0                 Type: OSD            Status: Ready
  Queue size: 0     CPU: 0     Address: 0540        Port name: OSD0PORT
  IPv4 Router Type: NonRouter  Arp Query Support: Yes
  IPv6 Router Type: NonRouter
    Link LINKOSD0              Type: QDIOETHERNET   Net number: 0
      Transport type: IP
      Speed: 100000000
      BytesIn: 33932           BytesOut: 15534
      Forwarding: Enabled      MTU: 8192            IPv6: Enabled
      IPv4 Path MTU Discovery: Disabled
      Broadcast Capability: Yes
      Multicast Capability: Yes
      IPv4 VIPA ARP
      IPv6 VIPA ND
      Multicast Group                        Members
      ---------------                        -------
      224.0.0.1                                 1
      FF02::1:FF00:1F                           1
      FF02::1                                   1
Ready;
```

In the preceding example, the first device in this example is DEVHS0, which is a device type of HIPERS (a HiperSocket device). This device has a device driver that runs on virtual CPU 0, a base virtual address of 0520, a port name of HSA0PORT, a IPv4 router type of NonRouter, and with ARP query capability. There is one link defined for this device, named LINKHS0, which is a device type of QDIOIP (Queued Direct I/O Internet Protocol) that has LAN broadcast capabilities and is multicast-capable. There is one member in the multicast group 224.0.0.1.

The second device indicated is DEVOSD0, which is a device type of OSD (indicating it is an OSA-Express Adapter using the QDIO Hardware Facility) at base virtual address 0540 with a port name of OSD0PORT and whose device driver runs on virtual CPU 0. This device is also an IPv4 and IPv6 NonRouter type with ARP query capability. One link is defined for this device, named LINKOSD0, which is a QDIOETHERNET device type that has LAN broadcast capabilities and is multicast-capable. IPv4 VIPA ARP indicates that this device is responsible for handling Address Resolution Protocol (ARP) requests for IPv4 VIPA addresses and Proxied IP addresses on this network. IPv6 VIPA ND indicates that this device is responsible for handling Neighbor Discovery requests for IPv6 VIPA addresses on this network. IPv4 VIPA ARP and IPv6 VIPA ND may be displayed even when VIPA addresses have not been configured on this system. In this case, they indicate the interface which would be responsible for VIPA addresses should they be added at a later time. There is one member in the IPv4 multicast group (224.0.0.1) and one member in each of the IPv6 multicast groups (FF02::1:FF00:1F and FF02::1). LINKOSD0 is enabled for IPv6 support.

DOS

The following is a sample of the information that is displayed after entering NETSTAT DOS.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Maximum Number of Half Open Connections: 2500
Maximum Number of Connections Per Foreign IP Address: 128
Maximum Number of Persist Connections: 2500


                             Attacks    Elapsed     Attack
Attack    IP Address        Detected       Time   Duration
--------  ---------------  ---------  ---------  ---------
Fraggle   10.130.248.97          2450    0:08:53    0:02:25
          10.130.248.99           100    0:05:31    0:00:50
Smurf-OB  10.130.58.253           120    0:18:21    0:00:34
          10.130.58.25            330    0:18:21    0:00:36
          10.32.232.45            165    0:16:43    0:02:32
          10.117.222.18             3    0:08:41    0:00:03
          10.130.249.43             2    0:08:15    0:00:00
          10.130.58.22              2    0:07:49    0:00:01
          10.117.32.30              5    0:06:45    0:01:41
          *                      1450    0:05:47    0:04:30
POD       10.130.58.22          23743    0:01:11    0:01:00
SSTRESS   10.130.58.23             97    22:50:02   0:00:45
Ready;
```

DROP

The following is a sample of the information that is displayed after entering NETSTAT DROP 1002.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Connection successfully dropped
Ready;
```

Note: You must be a privileged user to use the DROP command.

FILTERS

The following is a sample of the information that is displayed after entering NETSTAT FILTERS.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Blocked IP addresses:
                           Packets           Active            Last
IP Address                 Discarded         For               Packet
----------                 ----------        -------           ------
9.130.48.223                    77           0:03:22           0:00:37
9.130.48.56                      9           0:03:22           0:01:21
Ready;
```

GATE

The following is a sample of the information that is displayed after entering NETSTAT GATE.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIPB6

Path MTU Discovery Aging Interval: 10 Minutes

Known IPv4 gateways:

Subnet Address  Subnet Mask      FirstHop        Flgs PktSz Metric Link
--------------  -----------      --------        ---- ----- ------ ------
Default         <none>           9.60.28.1       UG   1500  0      TOVSL2SWA6
9.60.28.0       255.255.255.128  <direct>        US   1500  1      TOVSL2SWA6
9.152.224.54    HOST             9.60.28.90      UGHS 1500  1      TOVSL2SWA6

Known IPv6 gateways:

NetAddress:   Default
  FirstHop:   FE80::9:57FF:FE60:10F
  Flags:      UGA                   PktSz:  1500
  Metric:     1
  Link:       TOVSL2SWA6
NetAddress:   50C6:A2C2::/64
  FirstHop:   <direct>
  Flags:      UA                    PktSz:  1500
  Metric:     2
  Link:       TOVSL2SWA6

Ready;
```

ROUTERADV: The following is a sample of the information that is displayed after entering NETSTAT GATE ROUTERADV:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIPB6

Gateways from Router Advertisements:

NetAddress:   Default
  FirstHop:   FE80::9:57FF:FE60:10F
  Flags:      UGA                   PktSz:  1500
  Preference: High
  Reachable:  Yes
  Link:       TOVSL2SWA6
NetAddress:   Default
  FirstHop:   FE80::D6:ABFF:FE00:37
  Flags:      UGA                   PktSz:  1500
  Preference: Medium
  Reachable:  Yes
  Link:       TOVSL2SWA6
NetAddress:   50C6:A2C2::/64
  FirstHop:   <direct>
  Flags:      UA                    PktSz:  1500
  Preference: Medium
  Link:       TOVSL2SWA6

Ready;
```

HELP

The following is a sample of the information that is displayed after entering NETSTAT HELP or NETSTAT ?.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Usage: netstat option/command modifier
Current information viewable:
ALL             - Everything about a connection
ALLCONN         - With CONN or INTERVAL options, shows
                  TIME-WAIT and CLOSED connections
ARP adr         - Query ARP entry
CLIENTS         - Current clients
CONFIG opts     - Query TCP/IP stack configuration.  NETSTAT CONFIG HELP
                  for more information
CONN            - Active transmission blocks
DOS             - Display denial of service attack information
FILTERS         - Display IP addresses being blocked
GATE            - Current known gateways
HOME            - Home address list
IDENTIFY adr    - Display connection information for an IP address
INTERVAL n      - Full screen, real-time, connection display
LEVEL           - TCP/IP software level information
NEIGHBOR adr    - Query neighbor cache entry for an IP address
POOLSIZE        - Free pool status
SOCKETS         - Socket interface users and their sockets
TCP server      - Displays detailed info about the specified TCP/IP server
TELNET          - Telnet connections and logical devices
UP              - Date and time VM TCP/IP was last started
Commands available:
BLOCK adr       - Ignore packets from an IP address
CP command      - Issue a CP command
DELARP adr      - Delete ARP cache entry for an IP address
DELNEIGHBOR adr- Delete neighbor cache entry for an IP address
DROP n          - Drop a TCP connection
OBEY stmt       - Change TCP/IP configuration
RESETDOS        - Clear denial of service attack information
RESETPOOL       - Reset record of pool informs sent
UNBLOCK adr     - Process packets from an IP address
( SELECT select-value1...select-value6
                   For ALL CLIENTS CONN GATE INTERVAL TELNET select specific info
                   select-value may be a partial string terminated by a '*'
Ready;
```

HOME

The following is a sample of the information that is displayed after entering NETSTAT HOME.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

IPv4 Home address entries:

Address         Subnet Mask      Link              VSWITCH
-------         -----------      ------            -------
9.130.240.135   255.255.255.0    TO240             <none>
9.130.198.12    <none>           ROSA              <none>
9.130.198.16    <none>           LINKEE            VSWITCH7

IPv6 Home address entries:

Link:           ROSA
  Address:      50C6:C2C1::9:130:198:12
    Flags:
    DAD State:  Address Passed DAD
    Origin:     HOME statement

Link:           ROSA
  Address:      FE80::6:296C:9D01:C00
    Flags:      Autoconfigured
    DAD State:  Address Passed DAD
    Origin:     Link-local

Link:           TO240
  Address:      50C0:C2C1::9:130:240:135
  Flags:
  DAD State:    Address Passed DAD
  Origin:       HOME statement

Link:           TO240
  Address:      FE80::200:0:D:4040
  Flags:        Autoconfigured
  DAD State:    Address Passed DAD
  Origin:       Link-local

Ready;
```

In the preceding sample NETSTAT HOME output, the following are definitions for the IPv6 related fields:

Link: The name of the link associated with this IPv6 HOME entry.

Address:
The IPv6 address for this IPv6 HOME entry.

Flags: Possible values for Flags are:

Autoconfigured
This IPv6 address was automatically created by TCP/IP as a result of a received prefix in a Router Advertisement, a prefix on a HOME statement, a prefix on a ROUTERADVPREFIX statement, or the IPv6 address is a link local address.

Autoconfigured, Deprecated
The preferred lifetime of the autoconfigured IPv6 address has expired.

DAD State:
The status of Duplicate Address Detection (DAD) processing for this IPv6 Home entry. Values for DAD State are:

Pending Start of Link
The link is not active.

In Progress
Duplicate Address Detection is in progress.

Address Passed DAD
This address has passed Duplicate Address Detection.

Interface ID Passed DAD
The Interface Identifier in this IPv6 address had previously passed Duplicate Address Detection.

Address Failed DAD
This address has failed Duplicate Address Detection.

Interface ID Failed DAD
The Interface Identifier in this IPv6 address had previously failed Duplicate Address Detection.

DAD Bypassed; Device IP table update successful
Duplicate Address Detection has been bypassed for this IPv6 address. The IPv6 address was successfully added to the IP table for the device.

DAD Bypassed; Device IP table update failed
Duplicate Address Detection has been bypassed for this IPv6 address. The IPv6 address could not be added to the IP table for the device. See the messages on the TCP/IP stack console for the reason why this address got this failure.

Unknown DAD state; IPv6 disabled
The link local address for this link failed Duplicate Address Detection, so IPv6 is disabled on this link.

Origin:
The origin for this IPv6 address. Values for Origin are:

HOME statement
This IPv6 address was created by specifying the address or prefix on a HOME statement.

Received Router Advertisement
This IPv6 address was created as a result of receiving a prefix in a Router Advertisement.

ROUTERADVPREFIX statement
This IPv6 address was created as a result of specifying this prefix on a ROUTERADVPREFIX statement.

Link-local
This IPv6 address is the link local address that is automatically created by TCP/IP.

IDENTIFY

The following is a sample of the information that is displayed after entering NETSTAT IDENTIFY 9.130.57.*:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP
9.117.32.29 23 9.130.57.37 1081 INTCLIEN L00E6 DIALED TO YVETTE 0200
9.117.32.29 23 9.130.57.37 1082 INTCLIEN L00D8 DIALED TO PVM 050C
9.117.32.29 23 9.130.57.37 1081 INTCLIEN L000B ENABLED
9.117.32.29 23 9.130.57.37 1081 INTCLIEN L0027 LOGON AS M1GR2S 0009
9.117.32.29 1501 9.130.57.110 2538 DSOSERV
9.117.32.29 1501 9.130.57.54 1465 DSOSERV
9.117.32.29 1501 9.130.57.81 1764 DSOSERV
```

The following is a sample of the information that is displayed after entering NETSTAT IDENTIFY SSL:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP
9.117.32.29 990  9.60.67.161 2676 FTPSERVE E SSL00001 TSTCERT1 SSLV3  SHA1  RC4  RSA
9.117.32.29 992  9.60.67.161 2658 INTCLIEN I SSL00001
9.117.32.29 23   9.60.67.161 2663 INTCLIEN E SSL00001 TSTCERT1 SSLV3  SHA1  RC4  RSA
9.117.32.29 1024 9.60.67.161 2658 SSL00001 I SSL00001 TSTCERT2 SSLV3  SHA1  RC4  RSA
    ; /etc/gskadm/keyring data/system01
9.117.32.29 1058 9.117.32.29 23 SSL00001
9.117.32.29 1024 9.60.67.161 2663 SSL00001
9.117.32.29 1059 9.117.32.29 990 SSL00001
9.117.32.29 1024 9.60.67.161 2676 SSL00001
```

Note: In the preceding example, the data for the connection authenticated using the TSTCERT2 certificate spans two lines, due to formatting restrictions. For actual command results, this line and the subsequent line that contains the /etc/gskadm/keyring data/system01 text, instead would be a single line of data.

INTERVAL

The following is a sample of the information that is displayed after entering NETSTAT INTERVAL. The INTERVAL parameter can be used on an IBM 3278 or 3279 display station, or at a terminal or workstation that is emulating an IBM 3278 or 3279 display station.

Note:

1. The Enter key performs the same function as PF 3 (Quit).
2. The INTERVAL option provides a full–screen interface for its output as opposed to line-mode.

```
  07/13/05           VM TCP/IP Real Time Network Monitor             14:17:59
             Bytes     Bytes Local                 Idle
User Id        Out        In  Port   State         Time      Foreign Socket
--------   -------   ------- ------ -----------   -------   ----------------------------
INTCLIEN   1977670   3640468 TELNET Established   0:00:00   9.60.67.187..32813
MPROUTE    4770372 112917660 520    UDP           0:00:04   *..*
SMTP             0         0 1024   UDP           0:00:11   *..*
INTCLIEN     28371       526 TELNET Established   0:00:11   9.60.66.172..2007
VMNFS        51456      3772 2049   UDP           0:00:32   *..*
INTCLIEN    117237      1495 TELNET Established   0:00:56   9.60.67.133..1287
INTCLIEN      1367       476 TELNET Established   0:01:08   9.60.66.182..3492
INTCLIEN     15338       374 TELNET Established   0:01:08   9.60.67.133..1665
MOYW1            1         1 46467  Established   0:01:24   50C0:C2C1::200:0:100:1F..1024
MOYW1            0         0 46467  Listen        0:01:24   *..*
INTCLIEN    339563     23355 TELNET Established   0:01:53   9.65.33.57..1487
INTCLIEN     22855       555 TELNET Established   0:02:05   9.60.67.187..32819
INTCLIEN     18044       317 TELNET Established   0:02:05   9.60.67.187..32912
INTCLIEN     37175      4822 TELNET Established   0:03:02   9.60.69.153..1064
INTCLIEN     47561      2096 TELNET Established   0:03:02   9.60.66.182..3385
INTCLIEN     24099       585 TELNET Established   0:03:02   9.60.67.187..32817
INTCLIEN     49354       632 TELNET Established   0:04:57   9.60.66.172..2046
INTCLIEN    712792     95071 TELNET Established   0:05:54   9.60.67.151..2090
BFISHING         0         0 8088   Listen        0:56:57   *..*
INTCLIEN         0         0 TELNET Listen        1:36:29   *..*
REXECD           0         0 REXEC  Listen        7:43:04   *..*
PORTMAP     101336    157624 PMAP   UDP           7:43:07   *..*
VMNFS            0         0 2049   Listen        7:43:10   *..*
FTPSERVE         0         0 FTP-C  Listen       13:16:55   *..*
PORTMAP          0         0 PMAP   Listen       13:25:34   *..*
REXECD           0         0 RSH    Listen       14:11:33   *..*
PERFSVM          0         0 8081   Listen       18:28:32   *..*
SMTP             0         0 SMTP   Listen       18:29:11   *..*
MPROUTE          1         0 1024   Established  18:29:11   127.0.0.1..1025
MISCSERV         0         0 9      UDP          18:29:15   *..*
MISCSERV         0         0 9      Listen       18:29:15   *..*
MISCSERV         0         0 7      UDP          18:29:15   *..*
MISCSERV         0         0 7      Listen       18:29:15   *..*
MISCSERV         0         0 17     UDP          18:29:15   *..*
MISCSERV         0         0 17     Listen       18:29:15   *..*
MISCSERV         0         0 13     UDP          18:29:15   *..*
MISCSERV         0         0 13     Listen       18:29:15   *..*
MISCSERV         0         0 37     UDP          18:29:15   *..*
MISCSERV         0         0 37     Listen       18:29:15   *..*


 TCP/IP stack: TCPIP
 Refresh interval: 20 seconds.                            TCBs in Use:29
 1=Usr 2=Sock 3=Quit 4=BOut 5=BIn 6=PFSet2 7=Up 8=Dwn 9=T/B 10=Rgtleft 11=Ip@ 12=Rfsh
```

LEVEL

The following is a sample of the information that is displayed after entering NETSTAT LEVEL.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

IBM 2064; z/VM Version 5 2.0, service level 0000 (64-bit), VM
TCP/IP level 640; RSU 0000 running TCPIP MODULE E2 dated 07/08/05 at 12:34
TCP/IP Module Load Address:  00C42000
Ready;
```

NEIGHBOR

The following is a sample of the information that is displayed after entering NETSTAT NEIGHBOR ALL:

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Querying NEIGHBOR cache for address *

Link:  TONETA                      LinkType:  QDIOETHERNET
Internet Address:  50C0:C2C1::9:60:59:14
Ethernet Address:  02-09-57-00-00-25    State:  Reachable
Type:  ROUTER                      AdvAsDefaultRoute:  NO

Link:  TONETA                      LinkType:  QDIOETHERNET
Internet Address:  FE80::209:5700:100:25
Ethernet Address:  02-09-57-00-00-25    State:  Reachable
Type:  ROUTER                      AdvAsDefaultRoute:  NO

Ready;
```

OBEY

The following is a sample of the response to entering NETSTAT OBEY START TR1.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

OBEY command response is: OK
OBEY return code = 0
Ready;
```

OSAINFO

The following is a sample of the information that is displayed after entering NETSTAT OSAINFO or NETSTAT OSAINFO (SELECT DEVB100 DEVB101 DEVB102:

```
VM TCP/IP Netstat Level 620 TCP/IP Server Name: TCPIPBX

Device DEVB100: data as of 06/06/10 09:49:28
  Layer 2
    VMAC address:                           02-09-57-60-00-63
    VLAN ID:                                1

  Layer 3
    IPv4 Address:
    --------------
    9.100.200.123
    9.100.200.124
    9.100.200.125

    IPv4 Multicast Address:                 MAC Address
    ----------------------                  -----------------
    224.0.0.1                               02-09-57-60-00-63
    224.0.0.5                               02-09-57-60-00-63

Device DEVB101:
  Data not available yet. Reissue the NETSTAT OSAINFO command.

Device DEVB102: data as of 06/06/10 08:45:20 *BUFFERED*
  Layer 2
    VMAC address:                           02-09-57-60-00-63
    VLAN ID:                                1
  Layer 3
    IPv4 Address:
    --------------
    9.100.200.123
    9.100.200.124
    9.100.200.125

    IPv4 Multicast Address:                 MAC Address
```

```
  ---------------------                   ----------------
  224.0.0.2                               02-09-57-60-00-65
  224.0.0.4                               02-09-57-60-00-65
```

- **For Layer 2**:

```
NETSTAT OSAINFO (SELECT DEVB100
VM TCP/IP Netstat Level 620 TCP/IP Server Name: TCPIPBX

Device DEVB100: data as of 06/06/10 09:49:28
    VMAC address:                         02-09-57-60-00-63
    VLAN ID:                              1
```

- **For Layer 3**:

```
NETSTAT OSAINFO (SELECT DEVB100
VM TCP/IP Netstat Level 620 TCP/IP Server Name: TCPIPBX

Device DEVB100: data as of 06/06/10 09:49:28
  IPv4 Address:
  ---------------
  9.100.200.123
  9.100.200.124
  9.100.200.125

  IPv4 Multicast Address:           .......  MAC Address
  ----------------------            ...... . -----------------
  224.0.0.1                         .......  02-09-57-60-00-63
  224.0.0.5
```

In this example, data is displayed for devices DEVB100 and DEVB102. Data displayed consists of layer 2 and layer 3 information, giving IP, IP multicast and MAC addresses.

Note that only data for device DEVB100 could be retrieved with the OSAINFO command, while data for DEVB101 and DEVB102 was not returned in time from the OSA device. However, buffered data was available for DEVB102, visible by string '*BUFFERED*' in the output. For DEVB101, no buffered data was available as apparently the OSA Address Table (OAT) for this device was not requested yet.

POOLSIZE

The following is a sample of the information that is displayed after entering NETSTAT POOLSIZE.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

TCPIP Free pool status:
Object       No. alloc     # free      Lo-water    Permit size
======       =========     ======      ========    ===========
ACB             5000         4955         4776          500
CCB              750          416          416           50
Dat buf         1200         1149         1097          240
Sm dat buf      5000         4837         4584          500
Tiny dat buf       0            0            0            1
Env             1250         1250         1132          125
Lrg env           75           74           66           15
RCB               50           50           50            3
SCB             2000         1947         1795          133
SKCB             256          221          210           17
TCB             5000         4816         4540          333
UCB              500          488          484           33
Add Xlate       1500         1478            1            5
IP Route        3000         2993            1           60
Foreign IP addr 113           113          112            6
Segment ACK  100000         99996        99899         5000
FPSP total locked pages: 215, Unused locked pages: 64
FPSP allocation threshold: 54000, Low-water mark: 0
TCPIP machine size: 90M, Pools: 59267K, Avail: 8744K, Max block: 8716K
Ready;
```

RESETPOOL

The following is a sample of the information that is displayed after entering NETSTAT RESETPOOL.

Note: You must be a privileged user to use the RESETPOOL command.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Function performed
Ready;
```

SOCKET

The following is a sample of the information that is displayed after entering NETSTAT SOCKET.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Socket interface status:

Name: MNASH      Subtask: INET6001   Path id: 2
  Socket: 3   Type: Stream    State: Listen        Flags: L    Conn: 1017
    BoundTo: *..8855
    ConnTo:  *..*
  Socket: 4   Type: Stream    State: Established   Flags:      Conn: 1015
    BoundTo: 50C6:C2C1::9:60:28:215..8855
    ConnTo:  50C6:C2C1::9:60:28:215..1044
  Socket: 5   Type: Stream    State: Established   Flags:      Conn: 1012
    BoundTo: 50C6:C2C1::9:60:28:215..8855
    ConnTo:  50C6:C2C1::9:60:28:215..1045
  Socket: 6   Type: Stream    State: Established   Flags:      Conn: 1007
    BoundTo: 50C6:C2C1::9:60:28:215..8855
    ConnTo:  50C6:C2C1::9:60:28:215..1046
  Socket: 7   Type: Stream    State: Established   Flags:      Conn: 1013
    BoundTo: 50C6:C2C1::9:60:28:215..8855
    ConnTo:  50C6:C2C1::9:60:28:215..1047
  Socket: 8   Type: Stream    State: Established   Flags:      Conn: 1008
    BoundTo: 50C6:C2C1::9:60:28:215..8855
    ConnTo:  50C6:C2C1::9:60:28:215..1048
Name: MAINT510   Subtask: INET4001   Path id: 5    Pending call: recv
  Socket: 3   Type: Stream    State: Established   Flags:      Conn: 1006
    BoundTo: 9.60.28.215..1049
    ConnTo:  9.60.28.215..7788
  Socket: 4   Type: Stream    State: Established   Flags:      Conn: 1004
    BoundTo: 9.60.28.215..1050
    ConnTo:  9.60.28.215..7788
  Socket: 5   Type: Stream    State: Established   Flags:      Conn: 1018
    BoundTo: 9.60.28.215..1051
    ConnTo:  9.60.28.215..7788
```

TELNET

The following is a sample of the information that is displayed after entering NETSTAT TELNET.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

IPv4 Telnet server status:

Conn Status     Foreign Host     B out      B in  Logical device status
---- ------     ------------     -----      ----  ---------------------
1118 Establshd  9.130.57.67      89606     10125  L0017 DIALED TO PVM      0503
1115 Establshd  9.82.1.118        1811       161  L00C1 ENABLED
1067 Listen     *                    0         0
1345 Establshd  9.130.58.10     881941   1016232  L00D1 LOGON  AS CIBULAMA 0009
1213 FIN-wait-2 9.185.67.151    162931       967

IPv6 Telnet server status:

Connection: nnnn          Status: Established
  Foreign Host: 50C6:C2C1::9:130:198:12
  Bytes out: nnnnn        Bytes in: nnnnn
  Logical Device Status: Lxxxx LOGON AS username 0010
Ready;
```

A connection in the listen state is always available for an incoming open request.

UNBLOCK

The following is a sample of the information that is displayed after entering NETSTAT UNBLOCK 9.130.48.*

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

Function performed
Ready;
```

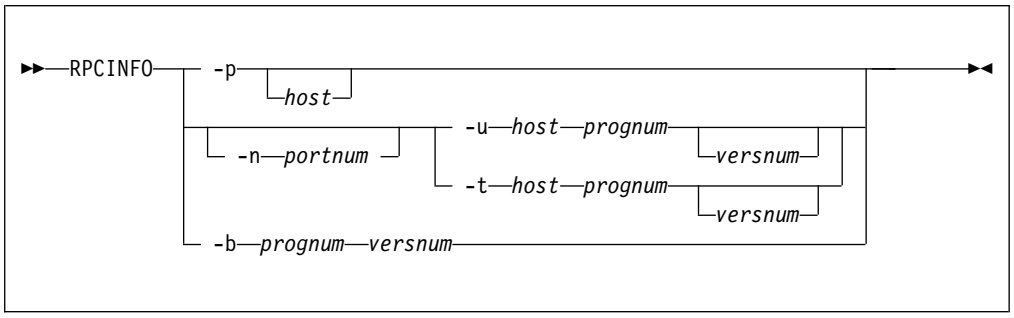
UP

The following is a sample of the information that is displayed after entering NETSTAT UP.

```
VM TCP/IP Netstat level 640       TCP/IP Server Name: TCPIP

TCP/IP started at 17:04:15 on 07/13/05
Ready;
```

RPCINFO Command



Purpose

Use the RPCINFO command to display the servers that are registered and operational with any portmapper on your network. The RPCINFO command makes a Remote Procedure Call (RPC) to an RPC server and displays the results.

Operands

-p *host*
: Queries the portmapper on the specified host and prints a list of all registered RPC programs. If *host* is not specified, the system defaults to the local host name.

-n *portnum*
: Specifies the port number to be used for the -t and -u options in place of the port number that is given by the portmapper.

-u *host prognum versnum*
: Sends an RPC call to procedure 0 of *prognum* on the specified host using UDP, and reports whether a response is received. The variable *prognum* is the name or number of the RPC program.

-t *host prognum versnum*
: Sends an RPC call to procedure 0 of *prognum* on the specified host using TCP, and reports whether a response is received.

-b *prognum versnum*
: Sends an RPC broadcast to procedure 0 of the specified *prognum* and *versnum* using UDP, and reports all hosts that respond.

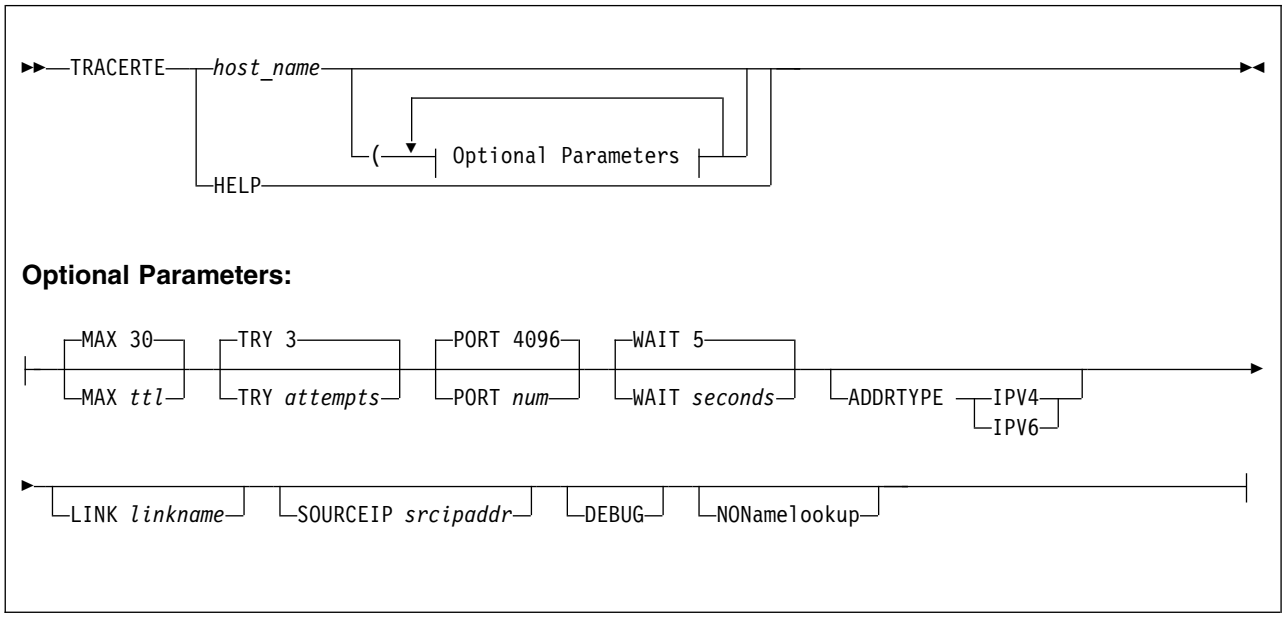
If you specify *versnum* (the version number), RPCINFO attempts to call that version of the specified program. If you do not specify a version, RPCINFO prints error information.

Note: The version number is required for the -b parameter.

Usage Notes

1. RPCINFO requires SCEERUN LOADLIB to be globally available. You should ensure the IBM Language Environment can be accessed. It is included in the z/VM base.

TRACERTE Command



Purpose

Use the TRACERTE command to debug network problems. The Traceroute function sends UDP requests with varying Time-to-live (TTL) and listens for TTL-exceeded messages from the routers between the local host and the foreign host.

Operands

host_name
Specifies the internet host name used to route packets.

MAX *ttl*
Specifies the maximum TTL. The range for valid values is 1 to 255. The default is 30.

TRY *attempts*
Specifies the number of attempts. The range for valid values is 1 to 20. The default is three.

PORT *num*
Specifies the starting port number. The range for valid values is 2048 to 60000. The default is 4096.

WAIT *seconds*
Specifies the number of seconds to wait for a response. The range for valid values is 1 to 255. The default is five.

ADDRTYPE IPV4
Indicates that TRACERTE is to attempt to resolve the specified host name to an IPv4 type address. If the ADDRTYPE option is not specified, TRACERTE will attempt to determine the address type to which the host name should be resolved based on a LINK option being specified, or on a SOURCEIP option being specified. If none of these options are specified, TRACERTE will first try to resolve the host name to an IPv6 address, and, if unsuccessful, will attempt to resolve the host name to an IPv4 address.

ADDRTYPE IPV6
Indicates that TRACERTE is to attempt to resolve the specified host name to an IPv6 type address. If the ADDRTYPE option is not specified, TRACERTE will attempt to determine the address type to which the host name should be resolved based on a LINK option being specified, or on a SOURCEIP option being specified. If none of these options are specified, TRACERTE will first try to resolve the host name to an IPv6 address, and, if unsuccessful, will attempt to resolve the host name to an IPv4 address.

LINK *linkname*
Allows you to specify the link on which to send out the packet. The LINK parameter is valid for tracing IPv6 targets only.

When issuing TRACERTE to a link-local address, the LINK option **must** be specified or the TRACERTE will fail. This is due to the fact that all IPv6 link-local addresses have the same prefix (FE80::Interface_ID), so, depending on how the Interface_ID is generated, hosts on different links could have the same link-local address. If the LINK option is not specified, the link to be used is determined from the routing table.

SOURCEIP *srcipaddr*
Allows you to specify the source IP address that is placed in the IP header when the packet is sent out. This is needed because IPv6 can have multiple IP addresses (link-local, site-local, and global) associated with a single link, and you need to verify that the TTL-exceeded error is returned to a specific address. If the SOURCEIP option is not specified, it is determined from the home list.

DEBUG
Specifies that extra messages are to be printed.

NONamelookup
Specifies that TRACERTE should print hop addresses numerically rather than both symbolically and numerically. This eliminates a name server query at each hop along the path.

Usage Notes

1. You can enter an optional parameter multiple times, but only the last instance is used.

   Example: In the following example, MAX is used twice, which is valid, but the program uses the last one (MAX 15):

   ```
   tracerte 9.60.67.166 ( MAX 25 MAX 15
   ```

2. The ADDRTYPE option allows you to specify the address type that should be returned when resolving the destination host name that was specified on the TRACERTE command. If the ADDRTYPE option is not specified, TRACERTE will attempt to determine the address type to which the host name should be

resolved based on a LINK option being specified, or on a SOURCEIP option being specified. If none of these options are specified, TRACERTE will first try to resolve the host name to an IPv6 address, and, if unsuccessful, TRACERTE will attempt to resolve the host name to an IPv4 address.

3. The range of port numbers that the TRACERTE command uses are normally invalid; however you can change the starting port number for this range if the target host is using a nonstandard UDP port.
4. The TRACERTE function will give unpredictable results if the TCP/IP stack is configured to use equal-cost multipath support. With this support, there may be multiple paths to a single destination. When TRACERTE is invoked, it will send the UDP requests out through the different paths. Since the packets will be traveling different paths to the same destination, the responses returned will not be for a single path.
5. TRACERTE uses the domain name server for inverse name resolution. If a host name is returned from the domain name server, it is printed along with its IP address. For more information about using domain name servers, see *z/VM: TCP/IP Planning and Customization*.
6. If TRACERTE receives an ICMP destination unreachable code other than "Port Unreachable," the TRACERTE command issues one of the following flags and stops processing:

Code	Meaning
!H	Destination address is not reachable
!N	Destination network is not reachable
!P	Destination protocol is not accessible
!S	Destination address is administratively blocked
!F	Fragmentation is needed

Return codes

0	Command Successful
8	Syntax Error
16	Socket Error
100	Command line is too long

Examples

The following are examples using the TRACERTE command:

- The second hop does not send TTL (Time-to-live) exceeded messages. Sometimes packets are lost (hops 6, 7, and 10).

```
tracerte cyst.watson.ibm.com
Trace route to CYST.WATSON.IBM.COM (9.2.91.34)
1 (9.67.22.2) 67 ms 53 ms 60 ms
2 * * *
3 (9.67.1.5) 119 ms 83 ms 65 ms
4 (9.3.8.14) 77 ms 80 ms 87 ms
5 (9.158.1.1) 94 ms 89 ms 85 ms
6 (9.31.3.1) 189 ms 197 ms *
7 * * (9.31.16.2) 954 ms
8 (129.34.31.33) 164 ms 181 ms 216 ms
9 (9.2.95.1) 198 ms 182 ms 178 ms
10 (9.2.91.34) 178 ms 187 ms *
```

- The network was found, but no host was found.

```
tracerte 129.35.130.09
Trace route to 129.35.130.09 (129.35.130.9)
1 (9.67.22.2) 61 ms 62 ms 56 ms
2 * * *
3 (9.67.1.5) 74 ms 73 ms 80 ms
4 (9.3.8.1) 182 ms 200 ms 184 ms
5 (129.35.208.2) 170 ms 167 ms 163 ms
6 * (129.35.208.2) 192 ms !H 157 ms !H
```

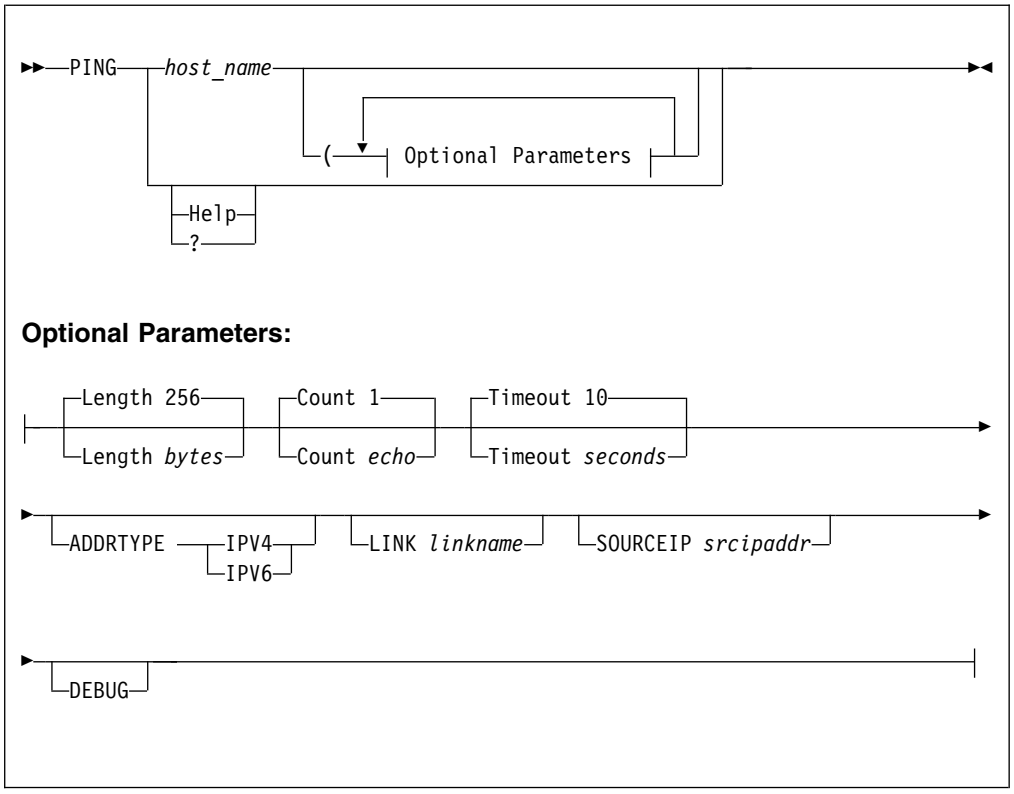
- Could not route to that network.

```
tracerte 129.45.45.45
Trace route to 129.45.45.45 (129.45.45.45)
1 (9.67.22.2) 320 ms 56 ms 71 ms
2 * * *
3 (9.67.1.5) 67 ms 64 ms 65 ms
4 (9.67.1.5) 171 ms !N 68 ms !N 61 ms !N
```

- Example with IPv6 addresses:

```
tracerte 50C6:C2C1:DFEC:ABCD:9:60:28:216
Trace route to 50C6:C2C1:DFEC:ABCD:9:60:28:216 (50c6:c2c1:dfec:abcd:9:60:28:216)
1   (50c6:c2c1::9:60:28:1) 2 ms 1 ms 3 ms
2   (50c6:c2c1::9:60:28:215) 2 ms 2 ms 2 ms
3   (50c6:c2c1:dfec:abcd:9:60:28:216) 2 ms 3 ms 2 ms
```

PING Command



Purpose

Use the PING command to send an echo request to a foreign node to determine if the node is accessible. PING uses ICMP as its underlying protocol.

Operands

host_name
Specifies the foreign host to which you want to send the echo request. If you omit the *host_name*, the system prompts you for a host name. The host name is either a character-string name or the internet address in the standard format of the foreign host (dotted-decimal format for IPv4 addresses, and either full or compressed format for IPv6 addresses).

Help
? Provides help information about the PING command. You cannot place the HELP parameter on the PING command line with other parameters.

Length *bytes*
Sets the number of bytes of the echo request. If you do not specify the operand, the default is 256. The number of bytes must be between 8 and the maximum value determined by large envelope size (*lrg_env_size*). For more information about *large_env_size*, see *z/VM: TCP/IP Planning and Customization*.

Count *echo*
Sets the number of echo requests that are sent to the foreign host. If you do not specify the operand, the default is 1. The number *echo* must be between 1 and 2^{31} -1 (2 147 483 647). If *echo* is 0, the PING command sends echo requests continually.

Timeout *seconds*
Sets the number of seconds that the PING command waits for a response. If you do not specify the operand, the default is 10. The number for *seconds* must be between 1 and 100.

ADDRTYPE IPV4
Indicates that PING is to attempt to resolve the specified host name to an IPv4 type address. If the ADDRTYPE option is not specified, PING will attempt to determine the address type to which the host name should be resolved based on a LINK option being specified, or on a SOURCEIP option being specified. If none of these options are specified, PING will first try to resolve the host name to an IPv6 address, and, if unsuccessful, it will attempt to resolve it to an IPv4 address.

ADDRTYPE IPV6
Indicates that PING is to attempt to resolve the specified host name to an IPv6 type address. If the ADDRTYPE option is not specified, PING will attempt to determine the address type to which the host name should be resolved based on a LINK option being specified, or on a SOURCEIP option being specified. If none of these options are specified, PING will first try to resolve the host name to an IPv6 address, and, if unsuccessful, it will attempt to resolve it to an IPv4 address.

LINK *linkname*
Allows you to specify which link to send the packet out on. The LINK parameter is valid for pinging IPv6 targets only.

If the LINK option is not specified, the link to be used is determined from the routing table. When issuing PING to a link-local address, the LINK option **must** be specified or the PING will fail. This is due to the fact that all IPv6 link-local addresses have the same prefix (FE80::Interface_ID), so, depending on how the Interface_ID is generated, hosts on different links could have the same link-local address.

SOURCEIP *srcipaddr*
Allows you to specify the source IP address that is placed in the IP header

when the packet is sent out. This is needed because IPv6 can have multiple IP addresses (link-local, site-local, and global) associated with a single link, and you need to verify that the PING is returned to a specific address. If the SOURCEIP option is not specified, it is determined from the home list.

DEBUG
Displays more detailed information about the packets being sent and received by the PING program.

Usage Notes

1. More than one option can be placed on the PING command line; however, the HELP parameter is an exception and cannot be placed on the PING command line with other parameters.
2. You can enter an optional parameter multiple times, but only the last instance is used.

   Example: In the following example, LENGTH is used twice, which is valid, but the program uses the last one (LENGTH 1000):

   ```
   ping 9.60.67.166 ( LENGTH 900 LENGTH 1000
   ```
3. When the TCP/IP stack is configured to use equal-cost multipath support, and there are multiple equal-cost paths to the destination being pinged, the ping results may seem inconsistent because the echo requests will be sent out through different interfaces. In the example below, If you ping 10.2.1.3 twice (or once with a COUNT 2 specified), the first echo request will be sent out through LINK1, and the second echo request will be sent out through LINK2.

   ```
   GATEWAY
   10.2.1.3     =     LINK1     4000     HOST
   10.2.1.3     =     LINK2     4000     HOST
   ```
4. PING does not support CMS subset mode.
5. If you issue HX while PING is running, the only valid command at the VM READ is BEGIN. Any other CMS command issued will be treated as if it is the BEGIN command. You will only see the `Ready;` message.

Return codes

0 Command Successful.

4 One or more of the PING attempts timed out.

8 A communication error between PING and the TCP/IP stack occurred.

16 Socket Error.

100 A PING command was issued with invalid syntax.

Examples

The following are examples of using the PING command:

- PING to an IPv4 destination:

  ```
  ping gdlvm7
  Ping level 640: Pinging host GDLVM7 (9.56.212.11).
                  Enter #CP EXT to interrupt.
  PING: Ping #1 response took 0.024 seconds. Successes so far 1.
  Ready; T=0.04/0.06 12:48:12
  ```
- PING to an IPv6 destination:

```
ping linuxipv62.tcp (addrtype ipv6
Ping level 640: Pinging host LINUXIPV62.TCP.raleigh.ibm.com
(fec0:0:0:1:9:67:114:44)
               Enter #CP EXT to interrupt.
PING: Ping #1 response took 0.002 seconds. Successes so far 1.
Ready; T=0.04/0.06 12:48:12
```

- PING to an IPv6 address:

```
ping 50C6:C2C1::9:60:28:215
Ping Level 510: Pinging host 50C6:C2C1::9:60:28:215.
                Enter #CP EXT to interrupt.
PING: Ping #1 response took 0.004 seconds. Successes so far 1.
Ready;
```

Chapter 8. SSL Certificate/Key Management and SSL Tracing Information

This topic discusses

- How to use the GSKKYMAN command to manage private SSL keys and certificates. In addition, detailed examples using the GSKKYMAN command are shown in this topic (see "GSKKYMAN Interactive Mode Examples" on page 247).
- How to manage certificate (key) database.
- How to set up SSL tracing.

Note: For more information about the principles of Certificate use in the SSL session, see Understanding Certificate Validation in *z/VM: TCP/IP Planning and Customization*.

SSL Certificate Management

SSL connections make use of public/private key mechanisms for authenticating each side of the SSL session and agreeing on bulk encryption keys to be used for the SSL session. To use public/private key mechanisms (termed PKI), public/private key pairs must be generated. In addition, X.509 certificates (which contain public keys) may need to be created, or certificates must be requested, received, and managed.

SSL support uses the **gskkyman** utility to manage PKI private keys and certificates. Invoke the **gskkyman** utility with the CMS GSKKYMAN command. GSKKYMAN creates, fills in, and manages a file that contains PKI private keys, certificate requests, and certificates. This file is called a **key database** and, by convention, has a file extension of **.kdb**.

SSL also uses PKCS #12 standard files created according to PKCS #12 V3.0. These files must be created as binary format files whose fully qualified file name does not exceed 251 characters in length and does not end with .kdb, .rdb, or .sth.

SSL supports PKCS #12 certificate and private key objects types. Any other object types within the file are ignored. All certificates within the file are treated as trusted certificates and no certificate can be identified as a default certificate.

The PKCS #12 file is protected by a password and the integrity of the file is ensured by a SHA-1 message authentication value.

When the certificates from a PKCS #12 file are read into storage they are assigned a label using either the PKCS #12 friendly name, if one exists, or the certificate's subject distinguished name. When the friendly name or the subject distinguished name value is greater than 127 characters, only the first 127 characters are used. If multiple certificates have the same friendly name value, the first encountered certificate is read into storage. Any other certificate with that friendly name is ignored. If a certificate is encountered that does not contain a friendly name and the subject distinguished name is empty, the processing of the PKCS #12 file fails. As with key database files, the label is case sensitive.

© Copyright IBM Corp. 1987, 2016

SSL uses the GSK_KEYRING_FILE environment variable to specify the locations of the PKI private keys and certificates. The key database file name or the PKCS #12 file name is passed in this environment variable.

gskkyman Overview

gskkyman is a program that creates, fills in, and manages a file that contains PKI private keys, certificate requests, and certificates. This file is called a **key database** and, by convention, has a file extension of .kdb. There is also an .rdb file that is a counterpart to the .kdb file.

The **gskkyman** utility only supports clear key operations.

The **gskkyman** utility only supports certificates that conform to RFC 2459: X.509 *certificate, certificate revocation list, and certificate extensions* or RFC 3280: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

RFC 5280: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* certificates can be used with **gskkyman** provided they conform to RFC 3280 rules for the certificate issuer name and subject name comparisons. Specifically, RFC 3280 indicates that UTF-8 values in the distinguished names must pass a case-sensitive (exact match) comparison to be considered equal. The **gskkyman** utility uses the issuer name and subject name values in the certificate to determine if a certificate is self-signed, and to perform certificate chaining. Therefore, **gskkyman** expects distinguished name attribute values to match according to a case-sensitive comparison when they are encoded as UTF-8 strings. Certificates that contain distinguished names with UTF-8 encoded attribute values for either the issuer name, the subject name, or both, that match through a case-insensitive comparison, can be created according to RFC 5280. Such certificates cause the **gskkyman** utility to fail checking for self-signed certificates and fail to correctly build certificate chains. Therefore, these certificates cannot be used with **gskkyman**.

The interface to **gskkyman**, while command-line based, is an interactive dialog between you (the user) and the program. At each step, the interactive **gskkyman** program prompts you with one or more lines of output and expects a numeric choice to be supplied as input at the prompt. Once a choice has been made, the **gskkyman** program prompts you for the individual pieces of information needed to fulfill the request. You are prompted for each piece of information. Many times there is a default choice that is listed between parentheses at the end of the command prompt. If the default choice is acceptable, press the space bar, then Enter, to select the default. If a choice other than the default is desired, enter the value at the prompt and press Enter. If a value is entered that is outside of the acceptable range of inputs, you will be re-prompted for the information.

Note: For a description of command line mode functions and options, see "GSKKYMAN Command Line Mode Syntax" on page 285.

Run **gskkyman** from CMS. You need access to the TCPMAINT 591 and 592 disks where the GSKKYMAN EXEC, load module, SSL code, and message catalogs reside. By default, **gskkyman** accesses the US English message catalogs. If you want **gskkyman** to access the Japanese message catalogs instead, set the NLSPATH environment variable by executing the following commands from a REXX EXEC:

```
nlspath = '/usr/lib/nls/msg/Ja_JP.IBM-939/%N'
'globalv select cenv put nlspath'
```

Key Database Files

Key database files are password protected because they contain the private keys that are associated with some of the certificates that are contained in the key database. Private keys, as their name implies, should be protected because their value is used in verifying the authenticity of requests made during PKI operations.

It is recommended that key database files be set with the following string of file permissions:

```
rw- --- --- (600) (read-write for only the owner of the key database)
```

The owner of the key database should be the user who will be managing the key database. The user ID that runs the program using the key database (LDAP or SSL server) must have at least read permission to the key database file at runtime. If the program runs under a different user ID than the administrator of the key database file , it is recommended that a group be setup to control access to the key database file. In this case, it is recommended that you set the permissions on the key database file to the following:

```
rw- r-- --- (640) (read-write for owner and read-only for group)
```

The owner of the key database file is set to the administrator user ID and the group owner of the key database file is set to the group that contains the server that will be using the key database file.

A key database that is created as a FIPS mode database, can only be updated by **gskkyman** or by using the CMS APIs executing in FIPS mode. Such a database, however, may be opened as read-only when executing in non-FIPS mode. Key databases created while in non-FIPS mode cannot be opened when executing in FIPS mode.

PKCS #12 Files

To use a PKCS #12 file in FIPS mode, the file must be protected using TDES. When creating a PKCS #12 file from certificates within a key database file, using the **gskkyman** utility, the key database must be a FIPS key database.

Elliptic Curve Cryptography Support

z/VM System SSL supports the following Elliptic Curve Cryptography (ECC) named curves:

- NIST recommended curves
  - secp192r1 – {1.2.840.10045.3.1.1}
  - secp224r1 – {1.3.132.0.33}
  - secp256r1 – {1.2.840.10045.3.1.7}
  - secp384r1 – {1.3.132.0.34}
  - secp521r1 – {1.3.132.0.35}
- Brainpool defined curves
  - brainpoolP160r1 – {1.3.36.3.3.2.8.1.1.1}
  - brainpoolP192r1 – {1.3.36.3.3.2.8.1.1.3}
  - brainpoolP224r1 – {1.3.36.3.3.2.8.1.1.5}
  - brainpoolP256r1 – {1.3.36.3.3.2.8.1.1.7}
  - brainpoolP320r1 – {1.3.36.3.3.2.8.1.1.9}
  - brainpoolP384r1 – {1.3.36.3.3.2.8.1.1.11}
  - brainpoolP512r1 – {1.3.36.3.3.2.8.1.1.13}

Note: In FIPS mode, only NIST recommended curves are currently supported. Curves under 224 bits are not recommended.

For data signature generation and verification operations involving ECC based algorithms, z/VM System SSL supports ECDSA with SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 digest algorithms. When creating signed certificates the recommended digest for the ECC key size of the signing private key is used (as specified in the following table).

Table 29. Recommended digest sizes for ECDSA signature key sizes

ECC curve type	ECDSA key sizes (bits)	Recommended digest algorithm	Signature algorithm type
x509_ecurve_brainpoolP160r1 x509_ecurve_secp192r1 x509_ecurve_brainpoolP192r1 x509_ecurve_secp224r1 x509_ecurve_brainpoolP224r1 x509_ecurve_secp256r1 x509_ecurve_brainpoolP256r1 x509_ecurve_brainpoolP320r1	160-383	SHA-256	x509_alg_ecdsaWithSha256
x509_ecurve_secp384r1 x509_ecurve_brainpoolP384r1	384-511	SHA-384	x509_alg_ecdsaWithSha384
x509_ecurve_brainpoolP512r1 x509_ecurve_secp521r1	512 and greater	SHA-512	x509_alg_ecdsaWithSha512

System SSL regards certain EC named curves to be the default curve for their key size. For ECC key generation that uses a key size parameter only, the default curve for the key size specified is used. These default EC named curves are outlined in the following table.

Table 30. Default EC Named Curves for Specified Key Sizes

Key size (bits)	Default EC named curve	Named curve OID
160	brainpoolP160r1	1.3.36.3.3.2.8.1.1.1
192	secp192r1	1.2.840.10045.3.1.1
224	secp224r1	1.3.132.0.33
256	secp256r1	1.2.840.10045.3.1.7
320	brainpoolP320r1	1.3.36.3.3.2.8.1.1.9
384	secp384r1	1.3.132.0.34
512	brainpoolP512r1	1.3.36.3.3.2.8.1.1.13
521	secp521r1	1.3.132.0.35

gskkyman Interactive Mode Descriptions

Interactive mode is entered when the GSKKYMAN command is entered without any parameters. A series of menus will be presented to allow you to select the database functions to be performed. Leading and trailing blanks will be removed from data entries but imbedded blanks will be retained. Blanks will not be removed from passwords.

Database Menu

This is the top-level menu and is displayed when the GSKKYMAN command starts:

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
```

Figure 13. Database Menu

Create new database

This option will create a new key database and the associated request database. You will be prompted to enter the key database name, the database password, the password expiration interval, and the database record length and choose either a FIPS or non-FIPS database (see "Key Database Files" on page 235 for a discussion of FIPS mode databases).

The fully-qualified key database name must be 2 to 251 characters. The file can contain an extension consisting of 1 to 3 characters. The recommended extension is ".kdb". If the name does not end with an extension to allow for the addition of an extension when creating the request database or the password stash file, the maximum database name is 247 characters. The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

The database password must be between 1 and 128 characters. A password exceeding 128 characters will be truncated to 128 characters.

The password expiration interval must be between 0 and 9999 days (a value of 0 indicates the password does not expire).

The record length must be large enough to contain the largest certificate to be stored in the database and must be between 2500 and 65536.

Two files will be created: the key database and the request database with an extension of '.rdb'. The file access permissions will be set so only the owner has access to the files.

Open database

This option will open an existing database. You will be prompted to enter the key database name and the database password.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

Change database password

This option will change the database password. You can change the password at any time but you must change it once it has expired in order to access the database once more. You will be prompted to enter the key

database name, the current database password, the new database password, and the new password expiration interval.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

The new database password must be between 1 and 128 characters.

The password expiration interval must be between 0 and 9999 days (a value of 0 indicates the password does not expire).

Change database record length

This option will change the database record length. All database records have the same length and database entries cannot span records. You can increase the record length if you find it is too small to store a new certificate. You can decrease the record length to reduce the database size if the original record length is too large. You cannot reduce the record length to a value smaller than the largest certificate currently in the database. You will be prompted to enter the key database name, the database password, and the new record length.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

The new record length must be between 2500 and 65536.

Delete database

This option will delete the key database, the associated request database, and the database password stash file. You will be prompted to enter the key database name.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

Create key parameter file

This option will create a file containing a set of key generation parameters. Key generation parameters are used when generating Digital Signature Standard (DSS) and Diffie-Hellman (DH) keys. The parameters will be stored in the specified file as an ASN.1-encoded sequence in Base64 format. This file can then be used when creating a signed certificate. The same key generation parameters can be used to generate multiple public/private key pairs. Using the same key generation parameters significantly reduces the time required to generate a public/private key pair. In addition, the Diffie-Hellman key agreement method requires both sides to use the same

group parameters in order to compute the key exchange value. Refer to FIPS 186-2 (Digital Signature Standard) and RFC 2631 (Diffie-Hellman Key Agreement Method) for more information on the key generation parameters. The key parameter generation process can take from 1 to 10 minutes depending upon key size, processor speed and system load.

Display certificate file (Binary or Base64 ASN.1 DER)
This option displays information about an X.509 certificate file. You will be prompted to enter the certificate filename. The fully-qualified certificate filename must be between 2 and 251 characters. The specified file must contain either a binary ASN.1 DER-encoded certificate or the Base64-encoding of a binary ASN.1 stream. A Base64-encoded certificate must be in the local code page.

Key Management Menu

The **Key Management Menu** is displayed once the key database has been created or opened. The key database and the associated request database are opened for update and remain open until you return to the **Database Menu**.

```
      Key Management Menu

      Database: Database_name
      Expiration: Expiration_date

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 14. Key Management Menu

Manage Keys and Certificates: This option manages certificates with private keys. A list of key labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

```
      Key and Certificate Menu

      Label: Certificate_label_name

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request


Enter option number (press ENTER to return to previous menu):
```

Figure 15. Key and Certificate Menu

Show certificate information
This option displays information about the X.509 certificate associated with the private key.

Show key information
This option displays information about the private key.

Set key as default
This option makes the current key the default key for the database.

Set certificate trust status
This option sets or resets the trusted status for the X.509 certificate. A certificate cannot be used for authentication unless it is trusted.

Copy certificate and key to another database
This option copies the certificate and key to another database. An error will be returned if the certificate is already in the database or if the label is not unique. A certificate and key may only be copied from a FIPS mode database to another FIPS mode database. A certificate and key may not be copied from a non-FIPS mode database to a FIPS mode database.

Export certificate to a file
This option exports just the X.509 certificate to a file. The supported export formats are ASN.1 Distinguished Encoding Rules (DER) and PKCS #7 (Cryptographic Message Syntax)

Export certificate and key to a file
This option exports the X.509 certificate and its private key to a file. The private key is encrypted when it is written to the file. The password you select will be needed when you import the file. The supported export formats are PKCS #12 Version 1 and PKCS #12 Version 3. The PKCS #12 Version 1 format is obsolete but is the only format supported by some SSL implementations. For FIPS mode databases, the export format supported is PKCS #12 Version 3. The strong encryption option uses Triple DES to encrypt the private key while the export encryption option uses 40-bit RC2. Strong encryption is the only supported option when exporting from a FIPS database. The export file will contain the requested certificate and its certification chain.

Delete certificate and key
The certificate and its associated private key are deleted.

Change label

This option will change the label for the database record.

Create a signed certificate and key

This option will create a new certificate and associated public/private key pair. The new certificate will be signed using the certificate in the current database record and then stored in the key database.

DSS and DH key generation parameters must be compatible with the requested key type and key size.

Keys are in the same domain if they have the same set of key generation parameters. Refer to FIPS 186-2 (Digital Signature Standard) and RFC 2631 (Diffie-Hellman Key Agreement Method) for more information on the key generation parameters. The subject name and one or more subject alternate names can be specified for the new certificate.

The subject name is always an X.500 directory name while a subject alternate name can be an X.500 directory name, a domain name, an e-mail address, an IP address, or a uniform resource identifier. An X.500 directory name consists of common name, organization, and country attributes with optional organizational unit, city/locality, and state/province attributes. A domain name is one or more tokens separated by periods. An e-mail address consists of a user name and a domain name separated by '@'. An IP address is an IPv4 address (nnn.nnn.nnn.nnn) or an IPv6 address (nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn). A uniform resource identifier consists of a scheme name, a domain name, and a scheme-specific portion.

The signature algorithm used when signing the certificate is derived from the key algorithm of the signing certificate and the following digest type:

- For RSA signatures, the digest type matches that used in the signature algorithm of the signing certificate. If the digest type is not a SHA-based digest, then SHA-1 will be used.
- For DSA signatures using a 1024-bit DSA key, the digest type will be SHA-1. When using a 2048-bit DSA key, the user is offered a choice of SHA-2 digest algorithms.
- For ECC Signatures, the digest type is the suggested digest for the key size of the ECC private key, as specified in Table 29 on page 236.

Possible signature algorithms are:

- x509_alg_sha1WithRsaEncryption
- x509_alg_sha224WithRsaEncryption
- x509_alg_sha256WithRsaEncryption
- x509_alg_sha384WithRsaEncryption
- x509_alg_sha512WithRsaEncryption
- x509_alg_dsaWithSha1
- x509_alg_dsaWithSha224
- x509_alg_dsaWithSha256
- x509_alg_ecdsaWithSha256
- x509_alg_ecdsaWithSha384
- x509_alg_ecdsaWithSha512

Create a certificate renewal request

This option will create a certification request using the subject name and public/private key pair from an existing certificate. The certificate request

will be exported to a file in Base64 format. This file can then be sent to a certification authority for processing. The certificate returned by the certification authority can then be processed using option 5 (Receive requested certificate or a renewal certificate) on the **Key Management Menu**. The new certificate will replace the existing certificate in the key database.

Manage Certificates: This option manages certificates without private keys. A list of key labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

```
      Certificate Menu

      Label: Certificate_label_name

   1 - Show certificate information
   2 - Set certificate trust status
   3 - Copy certificate to another database
   4 - Export certificate to a file
   5 - Delete certificate
   6 - Change label

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 16. Certificate Menu

Show certificate information
This option displays information about the X.509 certificate.

Set certificate trust status
This option sets or resets the trusted status for the X.509 certificate. A certificate cannot be used for authentication unless it is trusted.

Copy certificate to another database
This option copies the certificate to another database. An error will be returned if the certificate is already in the database or if the label is not unique. A certificate and key may only be copied from a FIPS mode database to another FIPS database. A certificate and key may not be copied from a non-FIPS mode database to a FIPS mode database.

Export certificate to a file
This option exports the X.509 certificate to a file. The supported export formats are ASN.1 DER (Distinguished Encoding Rules) and PKCS #7 (Cryptographic Message Syntax). The export file will contain just the requested certificate when the DER format is selected. The export file will contain the requested certificate and its certification chain when the PKCS #7 format is selected.

Delete certificate
The certificate is deleted.

Change label
This option will change the label for the database record.

Manage Certificate Requests: This option manages certificate requests. A list of request labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

```
      Request Menu

       Label: label_name

   1 - Show key information
   2 - Export certificate request to a file
   3 - Delete certificate request and key
   4 - Change label

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 17. Request Menu

Show key information
This option displays information about the private key associated with the certificate request.

Export certificate request to a file
This option exports the certificate request to a file in Base64 format. This file can then be sent to a certification authority for processing.

Delete certificate request and key
The certificate request and its associated private key are deleted.

Change label
This option will change the label for the database record.

Create New Certificate Request: This option will create a certificate request using either RSA encryption or DSA for the public and private keys. The certificate request will be exported to a file in Base64 format. This file can then be sent to a certification authority for processing.

The label has a maximum length of 127 characters and is used to reference the certificate in the request database. The label will also be used when the certificate is received, so it must be unique in both the request and key databases. It must consist of characters which can be represented as 7-bit ASCII characters (letters, numbers, and punctuation) in the ISO8859-1 code page.

The subject name and one or more subject alternate names can be specified for the new certificate. The subject name is always an X.500 directory name while a subject alternate name can be an X.500 directory name, a domain name, an e-mail address, an IP address, or a uniform resource identifier. An X.500 directory name consists of common name, organization, and country attributes with optional organizational unit, city/locality, and state/province attributes. A domain name is one or more tokens separated by periods. An e-mail address consists of a user name and a domain name separated by '@'. An IP address is an IPv4 address (nnn.nnn.nnn.nnn) or an IPv6 address (nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn). A uniform resource identifier consists of a scheme name, a domain name, and a scheme-specific portion (for example, http://www.endicott.ibm.com/main.html).

Receive Requested Certificate or a Renewal Certificate: This option will receive the signed certificate returned by the certification authority. The certificate can be either a new or renewal certificate issued in response to a certificate request or a renewal of an existing certificate without a corresponding certificate request. If the certificate was issued in response to a certificate request, the certificate request must still be in the request database. If this is a renewal certificate without a certificate request, the old certificate must still be in the key database and must

have the same issuer name and public key. If the key database does not contain the private key of the old certificate or contains certificates signed by the old certificate, then the subject name must also be the same when renewing the certificate.

The certificate file must contain either an ASN.1 DER-encoded sequence as defined in RFC 2459: X.509 certificate, certificate revocation list, and certificate extensions, RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, or a signed data message as defined in PKCS #7 (Cryptographic Message Syntax). The data can either be the binary value or the Base64 encoding of the binary value.

If the import file is in PKCS #7 format, the first certificate in the file must be the request certificate, otherwise the request will fail with 'unable to locate matching request'. The certification chain will be imported if it is contained in the import file. The certificate subject name will be used as the label for certificates added from the certification chain. A chain certificate will not be added to the database if the label is not unique or if the certificate is already in the database.

Base64 data is in the local code page. A DER-encoded sequence must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'. A PKCS #7 signed data message must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----' or start with the encoding header '-----BEGIN PKCS #7 SIGNED DATA-----' and end with the encoding footer '-----END PKCS #7 SIGNED DATA-----'.

An intermediate certificate is a certificate signed by another entity. The key database must already contain a certificate for the issuer. The certificate will not be imported if the certificate authenticity cannot be validated or if the database already contains the certificate.

The request database entry will be deleted once the certificate has been received.

Create a Self-Signed Certificate: This option will create a self-signed certificate using either RSA, DSA, or ECC encryption for the public and private keys and a certificate signature based on a SHA digest algorithm. The SHA digest algorithm that is used depends on the key algorithm that is chosen for the certificate:

- If an RSA certificate is requested, the user is prompted to choose the SHA digest algorithm required.
- An ECC certificate uses the suggested digest for the key size of the ECC key, as specified in Table 29 on page 236.
- A 1024-bit DSA certificate uses SHA-1. For a 2048-bit DSA certificate, the user is prompted to choose the SHA digest algorithm required.

Possible signature algorithms are:

- x509_alg_sha1WithRsaEncryption
- x509_alg_sha224WithRsaEncryption
- x509_alg_sha256WithRsaEncryption
- x509_alg_sha384WithRsaEncryption
- x509_alg_sha512WithRsaEncryption
- x509_alg_dsaWithSha1
- x509_alg_dsaWithSha224

- x509_alg_dsaWithSha256
- x509_alg_ecdsaWithSha256
- x509_alg_ecdsaWithSha384
- x509_alg_ecdsaWithSha512

The certificate can be created for use by a certification authority or an end user. A CA certificate can be used to sign other certificates and certificate revocation lists while an end user certificate can be used for authentication, digital signatures, and data encryption.

The label has a maximum length of 127 characters and is used to reference the certificate in the request database. The label is also used when the certificate is received, so it must be unique in both the request and key databases. It must consist of characters that can be represented as 7-bit ASCII characters (letters, numbers, and punctuation) in the ISO8859-1 code page.

The number of days until the certificate expires must be between 1 and 9999.

The subject name and one or more subject alternate names can be specified for the new certificate. The subject name is always an X.500 directory name while a subject alternate name can be an X.500 directory name, a domain name, an e-mail address, an IP address, or a uniform resource identifier. An X.500 directory name consists of common name, organization, and country attributes with optional organizational unit, city/locality, and state/province attributes. A domain name is one or more tokens separated by periods. An e-mail address consists of a user name and a domain name separated by '@'. An IP address is an IPv4 address (nnn.nnn.nnn.nnn) or an IPv6 address (nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn). A uniform resource identifier consists of a scheme name, a domain name, and a scheme-specific portion (for example, http://www.endicott.ibm.com/main.html).

Note: A self-signed end-entity certificate (server or client certificate) is not suggested for use in production environments and should only be used to facilitate test environments before production. Self-signed certificates do not imply any level of security or authenticity of the certificate because, as their name implies, they are signed by the same key that is contained in the certificate. However, certificates that are signed by a certificate authority indicate that, at least at the time of signature, the certificate authority approved the information that is contained in the certificate.

Import a Certificate: This option will add the contents of the import file to a key database file. The import file may contain one or more certificates without private keys. When each certificate is added to the key database, it is marked as trusted. The expiration date associated with each certificate cannot exceed February 6, 2106.

When adding certificates from the import file to a FIPS key database file only certificates signed with FIPS signature algorithms using FIPS-approved key sizes may be imported. When processing a chain of certificates, processing of the chain will terminate if a non-FIPS certificate is encountered. Certificates processed prior to the failing certificates will be added to the key database file. It is the responsibility of the importer to ensure that the file came from a FIPS source in order to maintain meeting FIPS 140–2 criteria.

The import file must contain either an ASN.1 DER-encoded sequence as defined in RFC 2459: X.509 certificate, certificate revocation list, and certificate extensions,

RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, or a signed data message as defined in PKCS #7 (Cryptographic Message Syntax). The data can either be the binary value or the Base64 encoding of the binary value.

If the import file is in PKCS #7 format, only the first certificate and its certification chain will be imported. The certificate subject name will be used as the label for certificates added from the certification chain. A certification chain certificate will not be added to the database if the label is not unique or if the certificate is already in the database.

Base64 data is in the local code page. A DER-encoded sequence must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'. A PKCS #7 signed data message must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----' or start with the encoding header '-----BEGIN PKCS #7 SIGNED DATA-----' and end with the encoding footer '-----END PKCS #7 SIGNED DATA-----'.

A root certificate is a self-signed certificate and will be imported as long as the certificate is not already in the key database.

An intermediate CA or end-entity certificate is a certificate signed by another entity. The key database must already contain a certificate for the issuer. The certificate will not be imported if the certificate authenticity cannot be validated or if the database already contains the certificate.

An existing certificate can be replaced by specifying the label of the existing certificate. The issuer name, subject name, and subject public key in the new certificate must be the same as the existing certificate. If the existing certificate has a private key, the private key is not changed when the certificate is replaced.

Import a Certificate and a Private Key: This option imports a certificate and the associated private key and adds it to the key database. The certificate will be marked as trusted when it is added to the database. When importing a certificate, the expiration date cannot exceed February 6, 2106.

The import file must contain an ASN.1 DER-encoded sequence as defined in PKCS #12 (Personal Information Exchange Syntax). The data can be either the binary value or the Base64 encoding of the binary value. Base64 data is in the local code page and must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'.

A root certificate is a self-signed certificate and will be imported as long as the certificate is not already in the key database.

An intermediate CA or end entity certificate is a certificate signed by another entity. The key database must already contain a certificate for the issuer. The certificate will not be imported if the certificate authenticity cannot be validated or if the database already contains the certificate.

Each certificate in the certification chain will be imported if it is present in the import file. The certificate subject name will be used as the label for certificates

added from the certification chain. A certification chain certificate will not be added to the database if the label is not unique or if the certificate is already in the database.

Only certificates and keys encoded according to PKCS #12 Version 3 and protected with strong encryption can be imported into a FIPS database. Furthermore, only certificates and keys comprising FIPS signature algorithms and using FIPS-approved key sizes may be imported into a FIPS database.

Show the Default Key: The private key information for the default key is displayed.

Store Database Password: The database password is masked and written to the key stash file. The file name is the same as the key database file name but has an extension of '.sth'.

Show Database Record Length: The database record length is displayed. All records in the database have the same length and a database entry cannot span a database record.

GSKKYMAN Interactive Mode Examples

gskkyman can be run from CMS. The following tasks will be performed in this section:

- Creating, opening and deleting a key database file
- Changing a key database password
- Storing an encrypted key database password
- Creating a self-signed server or client certificate
- Creating a certificate request and processing the signed request
- Creating a certificate to be used with Diffie-Hellman key exchange
- Managing keys and certificates:
  - Show certificate/key information
  - Marking a certificate (and private key) as the default certificate for the key database
  - Copying a certificate (and private key) to a different key database:
    - Copying a certificate without its private key
    - Copying a certificate with its private key
    - Copying a certificate with its private key to a key database on the same system
  - Removing a certificate (and private key) from a key database
  - Changing a certificate label
- Importing a certificate from a file as a trusted CA certificate
- Importing a certificate from a file with its private key
- Using GSKKYMAN to be your own certificate authority (CA)

Starting GSKKYMAN

To start **gskkyman**, enter `gskkyman` at the command prompt (see Figure 18 on page 248).

Note: In the examples that follow, your input is shown in **bold**, and places where you press the Enter key are noted with **<enter>**.

Figure 18 shows the GSKKYMAN start menu.

```
gskkyman <enter>

       Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
```

Figure 18. Starting Menu for GSKKYMAN

From the **Database Menu** for GSKKYMAN, you can create a new key database, open an existing key database, display the contents of a certificate file, change a database password, change a database record length, delete a database, or exit GSKKYMAN.

Creating, Opening and Deleting a Key Database File

To create a new key database, enter **1** at the command prompt on the **Database Menu**:

```
       Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
1 <enter>

Enter key database name (press ENTER to return to menu):
mykey.kdb <enter>
Enter database password (press ENTER to return to menu):
<enter password>
Re-enter database password:
<enter password>

Enter password expiration in days (press ENTER for no expiration):
35 <enter>

Enter database record length (press ENTER to use 5000):
<enter>
Enter 1 for FIPS mode database or 0 to continue:
1 <enter>

Key database /home/sufwl1/ssl_cmd/mykey.kdb created.

Press ENTER to continue.
```

Figure 19. Creating a New Key Database

Figure 19 shows the input prompts that GSKKYMAN produces when you choose **1** to create a new key database. As you can see, default choices are listed in

parentheses. In the example, by pressing the Enter key at the **Enter database record length** prompt, the default of 5000 was chosen.

Note:

1. When dealing with certificates that might be large in size or have large key sizes, for example 2,048 or 4,096, an initial key record length of 5,000 might be required.
2. The maximum length of the password specified for a key database file is 128 characters.
3. When creating a new key database file, you will be prompted whether you want a FIPS or non-FIPS database file created. For more information about FIPS mode databases, see "Key Database Files" on page 235.

After entering the database record length, a message displays confirming that your database was created (see Figure 19 on page 248). You are prompted to press Enter to continue. Doing so displays the **Key Management Menu** for the database you have created:

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 20. Key Management Menu for GSKKYMAN

Figure 20 shows the **Key Management Menu**. Entering **0** at this prompt exits GSKKYMAN. Pressing Enter at the prompt returns you to the **Database Menu**.

To open an existing key database file, on the **Database Menu**, enter option number **2** (see Figure 21 on page 250). You are then prompted for the key database name and password.

Note: Do not lose the key database password. There is no method to reset this password if you lose or forget the password. If the password is lost, the private keys stored in the key database are inaccessible, therefore, unusable.

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
2 <enter>
Enter key database name (press ENTER to return to menu):
mykey.kdb <enter>
Enter database password (press ENTER to return to menu):
<enter password>
```

Figure 21. Opening an Existing Key Database File

The key database name is the file name of the key database. The input file name is interpreted relative to the current directory when GSKKYMAN is invoked. You may also specify a fully qualified key database name.

After you enter the key database name and password, the **Key Management Menu** displays for the database you have selected to open, (see Figure 22).

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 22. Key Management Menu

To delete an existing database, from the **Database Menu**, select option **5** (see Figure 23 on page 251):

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
5 <enter>
Enter key database name (press ENTER to return to menu):
mykey.kdb <enter>

Enter 1 to confirm delete, 0 to cancel delete:
1 <enter>

Key database /home/sufwl1/ssl_cmd/mykey.kdb deleted.

Press ENTER to continue.
```

Figure 23. Deleting an Existing Key Database

You are prompted to enter the key database name that you wish to delete. Then you must enter **1** to confirm the delete, or **0** to cancel the delete. If you choose **1**, a message displays to confirm the file has been deleted.

Note: If you delete an existing key database, the associated request database and database password stash file (if existent) **will also be deleted**. It's important to note that anyone with write access to a key database can delete that database either by removing it with the **rm** command or by using GSKKYMAN subcommand.

Changing a Key Database Password

You can change a key database password. From the **Database Menu**, select option **3**:

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
3 <enter>

Enter key database name (press ENTER to return to menu):
mykey.kdb <enter>
Enter database password (press ENTER to return to menu):
<enter current password>
Enter new database password (press ENTER to return to menu):
<enter new password>
Re-enter database password:
<enter new password>
Enter password expiration in days (press ENTER for no expiration):
<enter>

Database password changed.

Press ENTER to continue.
```

Figure 24. Changing a Key Database Password

Figure 24 displays the prompts you are given. You first enter your current password. Then you select a new password, and enter it again to confirm. You can choose your password expiration in days or press Enter to have no expiration. A message displays to confirm the transaction.

Storing an Encrypted Key Database Password

In order for applications to use the key database file, the application must specify both the file name as well as its associated password. The password can either be specified directly or through a stash file containing the encrypted password. The stash file provides a level of security where the password does not have to be explicitly specified. To save the encrypted key database password, enter option **10** from the **Key Management Menu**:

Note: In the following task descriptions, it is assumed that you have opened the key database and are displaying the **Key Management Menu** panel.

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
10 <enter>

Database password stored in /home/sufwl1/ssl_cmd/mykey.sth.

Press ENTER to continue.
```

Figure 25. Storing a Database Password in a Stash File

Figure 25 shows the message you receive after entering option **10** to store the database password. In this example, the database password was stored in a file called `mykey.sth`.

Creating a Self-Signed Server or Client Certificate

If your organization does not use a certificate authority (within the organization or outside the organization), a self-signed certificate can be generated for use by the program acting as an SSL server or client. In addition, since root CA certificates are also self signed certificates that are permitted to be used to sign other certificates (certificate requests), these procedures can also be used to create a root CA certificate. See "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264.

Programs acting as SSL servers (i.e. acting as the server side of the SSL handshake protocol) must have a certificate to use during the handshake protocol. A program acting as an SSL client requires a certificate when the SSL server requests client authentication as part of the SSL handshake.

Note: This is not recommended for production environments and should only be used to facilitate test environments prior to production. Self-signed certificates do not imply any level of security or authenticity of the certificate because, as their name implies, they are signed by the same key that is contained in the certificate. On the other hand, certificates that are signed by a certificate authority indicate that, at least at the time of signature, the certificate authority approved the information contained in the certificate.

Note: GSKKYMAN supports the creation of X.509 Version 3 certificates.

When creating a self-signed certificate to be used to identify a server or client, from the **Key Management Menu**, enter **6**. You will be prompted for a number of items to define the certificate, including the intended use of the certificate, the key algorithm and key size, and possibly the digest algorithm for the certificate signature.

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
6 <enter>
```

Figure 26. Select 6 to Create a Self-Signed Certificate

Certificates that are intended to be used directly by a server or client are considered to be end-user certificates. Certificates intended to be used to sign other certificates are considered to be CA certificates. RSA key certificates are the most common. DSA key certificates represent certificates that follow the FIPS-186 government standard. ECC key certificates represent certificates that use Elliptic Curve Cryptography. The larger the key size, the more secure the generated key will be. Note that CPU usage increases as the key size increases.

If an RSA-based certificate is selected, you will be prompted to select the digest type for the signature algorithm from a list of SHA-based digest types. See Figure 27 on page 255 for an example of selecting the key size and digest type.

If a 1024-bit DSA certificate is selected, SHA-1 will be used for the signature algorithm. If a 2048-bit DSA certificate is selected, you will be prompted to select the digest type for the signature algorithm from a list of SHA-based digest types.

If an ECC certificate is selected, you will be prompted to select the ECC key type and curve type. The suggested digest for the key size of the ECC key will be used for the signature algorithm, as specified in Table 29 on page 236. See "Creating a Signed ECC Certificate and Key" on page 273 for more information.

Once the certificate type is determined, you will be prompted to enter:

- a label to uniquely identify the key and certificate within the key database
- the individual fields within the subject name
- certificate expiration. The valid expiration range is 1 to 9999 days. The default value is 365 days.
- the subject alternate names (optional)

Figure 27 on page 255 shows the creation of a self-signed certificate to be used as a server or client certificate.

```
       Certificate Usage

   1 - CA certificate
   2 - User or server certificate

Select certificate usage (press ENTER to return to menu):
2 <enter>

       Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu):
1 <enter>

       RSA Key Size

   1 - 1024-bit key
   2 - 2048-bit key
   3 - 4096-bit key

Select RSA key size (press ENTER to return to menu):
1 <enter>

       Signature Digest Type

   1 - SHA-1
   2 - SHA-224
   3 - SHA-256
   4 - SHA-384
   5 - SHA-512

Select digest type (press ENTER to return to menu):
1 <enter>

Enter label (press ENTER to return to menu):
Server Cert <enter>
Enter subject name for certificate
  Common name (required):
My Server Certificate <enter>

  Organizational unit (optional):
ID <enter>

  Organization (required):
IBM <enter>

  City/Locality (optional):
Endicott <enter>

  State/Province (optional):
NY <enter>

  Country/Region (2 characters - required):
US <enter>

Enter number of days certificate will be valid (default 365):
244 <enter>

Enter 1 to specify subject alternate names or 0 to continue:
0 <enter>

Please wait .....

Certificate created.

Press ENTER to continue.
```

Figure 27. Creating a Self-Signed Certificate

Once the certificate is created, the next step is to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by SSL without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264.

In order for the SSL handshake to successfully validate the use of the self-signed certificates, the partner application needs to know about the signer of the certificate. For self-signed certificates, this means the self-signed certificate must be imported into the partner's database. For more information on importing certificates, see "Importing a Certificate from a File as a Trusted CA Certificate" on page 278.

Setting Permission for the Certificate (Key) Database

1. Issue the **OPENVM** commands that follows to confirm that the necessary database files have been created, and to list the permissions of these files.

   openvm list /etc/gskadm/

```
Directory = '/etc/gskadm/'
Update-Dt  Update-Tm Type  Links          Bytes Path name component
09/12/2008 18:50:47   F        1          65080 'Database.kdb'
09/12/2008 18:51:21   F        1             80 'Database.rdb'
09/12/2008 18:51:01   F        1            129 'Database.sth'
Ready; T=0.01/0.01 18:51:34
```

   openvm list /etc/gskadm/ (own

```
Directory = '/etc/gskadm/'
User ID    Group Name  Permissions Type  Path name component
gskadmin   security    rw- --- ---  F    'Database.kdb'
gskadmin   security    rw- --- ---  F    'Database.rdb'
gskadmin   security    rw- --- ---  F    'Database.sth'
Ready; T=0.01/0.01 18:51:37
```

2. Issue the **OPENVM PERMIT** commands that follow to allow the SSL server to access the newly-created key database:

   openvm permit /etc/gskadm/Database.kdb rw- r-- ---

   openvm permit /etc/gskadm/Database.sth rw- r-- ---

3. Issue the **OPENVM LIST** command that follows to confirm that **r** (read) has been added to the "group" permissions for the key database and password stash files:

   openvm list /etc/gskadm/ (own

```
Directory = '/etc/gskadm/'
User ID    Group Name  Permissions Type  Path name component
gskadmin   security    rw- r-- ---  F    'Database.kdb'
gskadmin   security    rw- --- ---  F    'Database.rdb'
gskadmin   security    rw- r-- ---  F    'Database.sth'
Ready; T=0.01/0.01 18:55:15
```

Creating a Certificate Request and Processing the Signed Request

A program may require a certificate, associated with itself, depending on what side of the SSL connection the program is running. This requirement also depends on whether ***client authentication*** is requested as part of the SSL handshake. Programs acting as SSL servers (act as the server side of the SSL handshake protocol) **must** have a certificate to use during the handshake protocol. A program acting as an

SSL client requires a certificate in the key database if the SSL server requests ***client authentication*** as part of the SSL handshake operation. The way in which certificates are used within an organization will determine whether you need to create a certificate request. If the organization chooses to use a certificate authority (within the organization or outside of the organization), then you must generate a certificate request.

To create a certificate request, enter **4** from the **Key Management Menu**.

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
4 <enter>
```

Figure 28. Select 4 to Create a New Certificate Request

When creating a certificate request, you are prompted to select the key algorithm and the key size for the certificate to be requested. RSA key certificates are the most common. DSA key certificates represent certificates that follow the FIPS-186 government standard. ECC key certificates represent certificates that use Elliptic Curve Cryptography. The larger the key size, the more secure the encryption/decryption generated key will be.

If an RSA-based certificate is selected, you are prompted to select the digest type for the signature algorithm from a list of SHA-based digest types.

If a 1024-bit DSA certificate is selected, SHA-1 is used for the signature algorithm. If a 2048-bit DSA certificate is selected, you are prompted to select the digest type for the signature algorithm from a list of SHA-based digest types.

If an ECC certificate is selected, you will be prompted to select the ECC key type and curve type. The suggested digest for the key size of the ECC key is used for the signature algorithm, as specified in Table 29 on page 236.

After the certificate type is determined, you will be prompted to enter:

- a request file name to store the certificate request
- a label to uniquely identify the certificate request within the key database
- the individual fields within the subject name
- the individual fields within the subject alternate name (optional).

The Certificate Key Algorithm menu appears:

```
      Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu):
1 <enter>

      RSA Key Size

   1 - 1024-bit key
   2 - 2048-bit key
   3 - 4096-bit key

Select RSA key size (press ENTER to return to menu):
1 <enter>

      Signature Digest Type

   1 - SHA-1
   2 - SHA-224
   3 - SHA-256
   4 - SHA-384
   5 - SHA-512

Select digest type (press ENTER to return to menu):
3 <enter>

Enter request file name (press ENTER to return to menu):
certreq.arm <enter>

Enter label (press ENTER to return to menu):
Test Server Cert <enter>

Enter subject name for certificate
  Common name (required):
Test Server <enter>

  Organizational unit (optional):
ID <enter>

  Organization (required):
IBM <enter>

  City/Locality (optional):
Endicott <enter>

  State/Province (optional):
NY <enter>

  Country/Region (2 characters - required):
US <enter>

Enter 1 to specify subject alternate names or 0 to continue:
0 <enter>

Please wait .....

Certificate request created.

Press ENTER to continue.
```

Figure 29. Creating a Certificate Request

Press option **0** to continue or option **1** to specify the subject alternate names. If option **1** is selected, the **Subject Alternate Name Type** menu appears:

```
 Subject Alternate Name Type

   1 - Directory name (DN)
   2 - Domain name (DNS)
   3 - E-mail address (SMTP)
   4 - Network address (IP)
   5 - Uniform resource identifier (URI)

Select subject alternate name type (press ENTER if name is complete):
1 <enter>

Enter subject name for certificate
  Common name (required):
Test server <enter>

  Organizational unit (optional):
IBM <enter>

  Organization (required):
IBM <enter>

  City/Locality (optional):
Endicott <enter>

  State/Province (optional):
NY <enter>

  Country/Region (2 characters - required):
US <enter>

       Subject Alternate Name Type

   1 - Directory name (DN)
   2 - Domain name (DNS)
   3 - E-mail address (SMTP)
   4 - Network address (IP)
   5 - Uniform resource identifier (URI)

Select subject alternate name type (press ENTER if name is complete):
<enter>

Please wait .....
```

Figure 30. Subject Alternate Name Type

When specifying subject alternate names, you are prompted for the type of the alternate name. After the alternate name type is determined, you will be prompted to enter:

- the individual fields within the subject name.

After the individual fields are completed, press enter to continue or select one of the subject alternate name types. Repeat the process.

Once the certificate request (and associated subject alternate names) is created, a file with the name you specified will exist in the current working directory or directory specified in the filename. If you choose to exit GSKKYMAN, the program ends. Otherwise, the **Key Management Menu** (see Figure 22 on page 250) displays, allowing additional operations to be performed.

The certificate request created is stored in a file that is in base64-encoded format. This format is what is typically required by certificate authorities that create certificates. The following is the contents of the file created by the steps performed in Figure 29 on page 258:

```
$ cat certreq.arm <enter>
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBnjCCAQcCAQAwXjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5ZMREwDwYDVQQH
EwhFbmRpY290dDEMMAoGA1UEChMDSUJNMQswCQYDVQQLEwJJRDEUMBIGA1UEAxML
VGVzdCBTZXJ2ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKzcdSVhSvOC
WYwapH3188F3H/lA7RE1IxQISSgW6q+xEV/1qVdEfROU8ln/pV1TMrphid6iMr74
CmW7AwTwzBQhoV+ZPtf4VFL6xG+vsQ34fVbMs4nUhoSyKTvVyoRVLJXV3g9atr66
rj9l+yRiIqnBIHpByeHRxiv1N7KTFurvAgMBAAGgADANBgkqhkiG9w0BAQUFAAOB
gQBnlce61dpJWH0t01XLsgdL/LcWLpyMTyL8ZOJxkjtg8SUmaoAXfJF1S+iNjhw+
NZaeNepUOAbFbDxYWbF23WfFbIzHcgpdPUh7fOhMWrRXRLzwCKLD0TNPqked2mVW
9eVzgQ1FkD9z73NHNz6aJAJMAY/cDU2Yqb+xcfh94/uvHA==
-----END NEW CERTIFICATE REQUEST-----
$
```

Figure 31. Contents of certreq.arm after Certificate Request Generation

Sending the Certificate Request

The certificate request file can either be transferred to another system (for example, ftp as an ASCII text file) and then transferred to the certificate authority or placed directly into a mail message sent to a certificate authority using cut-and-paste methods.

In addition to the certificate request file that is generated, a request database (.rdb) file is also created or altered. The request database file will be named the same as the key database file, except it will have an extension of .rdb. For example, a key database file of key.kdb will cause a request database file of key.rdb to be created. This request database file must be saved along with the key database in order for the response for the certificate request to be successfully processed.

The certificate request must not be deleted from the database while the request is being processed by the signing certificate authority. The database certificate request is required for applicable processing when the signed certificate from the certificate authority is received. The removal of the certificate request from the database causes the private key associated with the certificate request to be lost.

Receiving the Signed Certificate or Renewal Certificate

Once a certificate is signed by the certificate authority in response to the certificate request, you must received it into the key database. This is for new certificates and renewal certificates.

To receive the certificate, you must store the Base64-encoded certificate in a file on the system to be read in by the GSKKYMAN command. This file should be in the current working directory when GSKKYMAN is started. If this file is on another working directory you will have to specify the fully qualified name.

Note: In order to receive the certificate the CA certificate must also exist in the key database. To store a CA certificate, refer to "Importing a Certificate from a File as a Trusted CA Certificate" on page 278.

To receive a certificate issued on your behalf, from the **Key Management Menu**, see Figure 22 on page 250 and enter option **5**.

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
5 <enter>

Enter certificate file name (press ENTER to return to menu):
signed.arm <enter>

Certificate received.

Press ENTER to continue.
```

Figure 32. Receiving a Certificate Issued for your Request

You are prompted for the name of the file that contains the Base64-encoded certificate that was returned to you by the certificate authority in response to a previously submitted certificate request (See "Creating a Certificate Request and Processing the Signed Request" on page 256). After receiving the certificate, you press Enter to continue working with the **Key Management Menu**. Upon completion of this step and prior to SSL using the certificate during the SSL handshake processing, you need to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by SSL without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264.

When received into a key database file, the certificate's expiration date should be monitored. When the expiration date is nearing (do not wait until it is expired), a new certificate should be obtained to replace the existing certificate. The new certificate can be a brand new certificate with new public private keys or a renewal certificate where existing keys and certificate information is used.

Managing Keys and Certificates

Once certificates are added to the key database, the following are some common operations that can be performed with the certificates.

- Show certificate/key information
- Mark a certificate (and private key) as the default certificate for the key database
- Export a certificate to a file or key database
- Remove a certificate (and private key) from a key database
- Change a certificate label
- Create a signed ECC certificate and key
- Create a certificate to be used with a fixed Diffie-Hellman key exchange
- Create a certificate renewal request.

Showing Certificate/Key Information: It is sometimes useful to display the information contained in the certificates that are stored in the key database. The information displayed includes, among others, the label, issuer/subject name, the version number of the certificate, the key size for the public/private key pair, and the expiration date.

To list information about certificates that contain private keys, from the **Key Management Menu** (see Figure 22 on page 250) select **1**, (Manage keys and certificates). This displays the **Key and Certificate List**.

```
      Key and Certificate List

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Test Server Cert
   2 - Server Cert

   0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
2 <enter>
```

Figure 33. Key and Certificate List

Select the number corresponding to the label for which you would like to display certificate/key information. The **Key and Certificate Menu** for the label you chose displays next (see Figure 34).

```
      Key and Certificate Menu

      Label: Server Cert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
1 <enter>
```

Figure 34. Key and Certificate Menu

On the **Database Menu**, you could choose **1** to display certificate information. This accesses the **Certificate Information** menu (see Figure 35 on page 263):

```
                    Certificate Information

               Label: Server Cert
           Record ID: 13
     Issuer Record ID: 13
             Trusted: Yes
             Version: 3
       Serial number: 3c73c6d0000e8076
         Issuer name: My Server Certificate
                      ID
                      IBM
                      Endicott
                      NY
                      US
        Subject name: My Server Certificate
                      ID
                      IBM
                      Endicott
                      NY
                      US
      Effective date: 2010/02/20
     Expiration date: 2015/10/22
  Signature algorithm: shalWithRsaEncryption
    Issuer unique ID: None
   Subject unique ID: None
 Public key algorithm: rsaEncryption
     Public key size: 1024
          Public key: 30 81 89 02 81 81 00 E5 19 BF 6D A3 56 61 2D 99
                      48 71 F6 67 DE B9 8D EB B7 9E 86 80 0A 91 0E FA
                      38 25 AF 46 88 82 E5 73 A8 A0 9B 24 5D 0D 1F CC
                      65 6E 0C B0 D0 56 84 18 87 9A 06 9B 10 A1 73 DF
                      B4 58 39 6B 6E C1 F6 15 D5 A8 A8 3F AA 12 06 8D
                      31 AC 7F B0 34 D7 8F 34 67 88 09 CD 14 11 E2 4E
                      45 56 69 1F 78 02 80 DA Dc 47 91 29 BB 36 C9 63
                      5C C5 E0 D7 2D 87 7B A1 B7 32 B0 7B 30 BA 2A 2F
                      31 AA EE A3 67 DA DB 02 03 01 00 01

 Number of extensions: 4

Enter 1 to display extensions, 0 to return to menu:
1 <enter>
```

Figure 35. Certificate Information

From the **Certificate Information** screen, you can also enter **1** to display certificate extensions:

```
      Certificate Extensions List

   1 - subjectKeyIdentifier
   2 - authorityKeyIdentifier
   3 - keyUsage (critical)
   4 - basicConstraints (critical)

Enter extension number (press ENTER to return to previous menu):
3 <enter>
```

Figure 36. Certificate Extensions List

Enter 3 on the Certificate Extensions List to show key usage information:

```
Certificate signature
CRL signature

Press ENTER to continue.
```

Figure 37. Key Usage Information

To display key information, from the **Key and Certificate Menu**, choose **2**, **Show Key Information**. This accesses the **Key Information** menu (see Figure 38):

```
                      Key Information

               Label: Server Cert
           Record ID: 13
     Issuer Record ID: 13
          Default key: Yes
 Private key algorithm: rsaEncryption
      Private key size: 1024
          Subject name: My Server Certificate
                        ID
                        IBM
                        Endicott
                        NY
                        US

Press ENTER to continue.
```

Figure 38. Key Information menu

Marking a Certificate (and Private Key) as the Default Certificate for the Key Database: Once a certificate has been added to the key database through either a certificate request or as a self-signed certificate, it can be marked as the default certificate for the key database. Marking a certificate as the default certificate allows it to be used by the programs calling SSL without having to explicitly supply the certificate's label.

To mark a certificate as the default certificate for the key database, from the **Key Management Menu** (see Figure 22 on page 250), choose **1**, (Manage keys and certificates), and on the **Key and Certificate List** (see Figure 33 on page 262, choose the label number you want to work with. The **Key and Certificate Menu** displays:

```
      Key and Certificate Menu

      Label: My Server Certificate

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
3 <enter>

Default key set.

Press ENTER to continue.
```

Figure 39. Marking a Certificate (and Private Key) as the Default Certificate

Choose **3** to set the certificate and private key as the default certificate for the key database.

Copying a Certificate (and Private Key) to a Different Key Database: After you have populated your key database file with certificates, it may be necessary for you to transfer a certificate to another key database on your system or a remote system. This transfer maybe necessary for the following reasons:

- The remote system or key database requires the signing certificate to be in its key database file for validation purposes. The certificate does not need to contain the private key information. These certificates are normally certificate authority (CA) certificates but may also be a self-signed certificate.
- The server or client certificate is being used by another application in a separate key database file.

Note: The source key database file and the target key database file must exist before the certificate can be copied to the target key database file. If the target is a FIPS database, then only a FIPS database can be the source.

Copying a Certificate Without its Private Key: To copy a certificate to a different key database format or to a different system without its private key (certificate validation), from the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label of the certificate to be copied and enter the number associated with the label. In the **Key and Certificate Menu**, enter option **6** to export the certificate to a file. The **Export File Format** menu appears:

```
      Export File Format

   1 - Binary ASN.1 DER
   2 - Base64 ASN.1 DER
   3 - Binary PKCS #7
   4 - Base64 PKCS #7

Select export format (press ENTER to return to menu):
1 <enter>

Enter export file name (press ENTER to return to menu):
expfile.der <enter>

Certificate exported.

Press ENTER to continue.
```

Figure 40. Copying a Certificate Without its Private Key

You are then prompted for what file format you would like for the exported certificate information. The file format is determined by the support on the receiving system.

After selecting the export format, you will be asked for a file name. You then will receive a message indicating that the certificate was exported. You can now transfer this file to the system and import the certificate into the key database file. If copying to a remote system, this file can now be transferred (in binary if option 1 or 3 has been selected or in ASCII (TEXT) if option 2 or 4 has been selected) to the remote system. For information on receiving the certificate into the key database file, see "Importing a Certificate from a File as a Trusted CA Certificate" on page 278). Upon successfully receiving the certificate, the certificate can now be used to validate the SSL's partner certificate. When the partner is the client, this means that the client can now validate the server's certificate and if the partner is the server, the server can validate the client's certificate when client authentication is requested.

You will also need to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by SSL without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264.

Copying a Certificate with its Private Key: To copy a certificate to a different key database format or to a different system with its private key, the certificate must be exported to a PKCS #12 formatted file. PKCS #12 files are password-protected to allow encryption of the private key information. from the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label of the certificate to be copied and enter the number associated with the label. In the **Key and Certificate Menu**, enter option **7** to export the certificate and private key to a file.

The **Export File Format** menu appears:

```
      Export File Format

    1 - Binary PKCS #12 Version 1
    2 - Base64 PKCS #12 Version 1
    3 - Binary PKCS #12 Version 3
    4 - Base64 PKCS #12 Version 3

Select export format (press ENTER to return to menu):
3 <enter>

Enter export file name (press ENTER to return to menu):
expfile.p12 <enter>

Enter export file password (press ENTER to return to menu):
<enter password>

Re-enter export file password:
<enter password>

Enter 1 for strong encryption, 0 for export encryption:
1 <enter>

Certificate and key exported.

Press ENTER to continue.
```

Figure 41. Copying a Certificate and Private key to a Different Key Database

You will then be prompted for what file format you would like for the exported certificate information.

The file format is determined by the support on the receiving system. In most cases the format to be used is Binary PKCS #12 Version 3. Export from a FIPS database must be PKCS #12 Version 3 using strong encryption.

After selecting the export format, you will be asked for a file name and password. You then will receive a message indicating that the certificate was exported. You can now transfer this file to the system and import the certificate into the key database file. If copying to a remote system, this file can now be transferred (in binary) to the remote system. For information on receiving the certificate into the key database file, see "Importing a Certificate from a File with its Private Key" on page 281). Upon successfully receiving the certificate, the certificate can now be used to identify the program. For example, the certificate can be used as the SSL server program's certificate or it can be used as the SSL client program's certificate.

Copying a Certificate with its Private Key to a Key Database on the Same System: To copy a certificate and its private key from one key database to another key database on the same system, you will need to know the target key database file name and password. If the source database is a FIPS database, then the target database must also be a FIPS database. If the source database is a non-FIPS database, then the target database must also be a non-FIPS database. From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label of the certificate to be copied and enter the number associated with the label. From the **Key and Certificate Menu**, enter **5** to copy a certificate and key to another database:

```
      Key and Certificate Menu

      Label: newimp

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press Enter to return to previous menu):
5 <enter>

Enter key database name (press Enter to return to previous menu):
target.kdb <enter>

Enter database password (press Enter to return to previous menu):
<enter password>

Record copied.

Press ENTER to continue.
```

Figure 42. Copying a Certificate with its Private Key to a Key Database on the Same System

You will then be prompted for the target key database name, and the target key database password. After the certificate is copied to the other key database file, you will receive a message indicating that the certificate has been successfully copied.

Note: When a certificate with a key marked as default is copied from a key database into another database, the certificate is not marked as the default key in that database.

Removing a Certificate (and Private Key) from a Key Database: You may want to remove a certificate from the key database. For example, you may want to remove a certificate after a certificate has expired and is no longer useful. Also, you may want to remove a certificate after a certificate has been exported to a different key database and is no longer needed as part of this key database.

Caution: Once you remove a certificate/private key pair from a key database, you cannot recover it unless it has previously been stored somewhere else (another key database file, a PKCS #12 file for certificate/private key pairs, or a DER-encoded or Base64-encoded file for just certificates). Be sure you no longer require the certificate (and private key if one is associated with the certificate in the key database) before you remove it from the key database.

From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label chosen when creating (or receiving) the certificate and enter the number associated with the label. In the **Key and Certificate Menu** (see Figure 43 on page 269), choose **8** to delete the certificate and key:

```
      Key and Certificate Menu

      Label: newimp

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
8 <enter>

Enter 1 to confirm delete, 0 to cancel delete:
1 <enter>

Record deleted.

Press ENTER to continue.
```

Figure 43. Delete Certificate and Key

Enter **1** to confirm the deletion of the certificate and key. A message appears, confirming that the record has been deleted. Once the certificate has been removed from the key database, it can no longer be used for identification or verification purposes by SSL during SSL handshake processing.

Changing a Certificate Label: From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label chosen when creating (or receiving) the certificate and enter the number associated with the label. In the **Key and Certificate Menu** (see Figure 44 on page 270), choose **9** to change the label:

```
      Key and Certificate Menu

      Label: cacert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
9 <enter>

Enter label (press ENTER to return to menu):
cacert2 <enter>

Label changed.

Press ENTER to continue.
```

Figure 44. Changing a Certificate Label

Enter the new label name and press Enter. A message confirms that the label name has been changed.

Creating a Signed Certificate and Key: Creating a signed certificate and key allows for a fastpath method for creating a signed certificate that resides in the same key database file as the displayed signing Certificate Authority certificate. From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. Find the label of the signing Certificate Authority certificate and enter the number associated with the label. In the **Key and Certificate Menu**, choose option **10** to create a signed certificate and key. This requires the displayed certificate to have signing capability.

```
      Key and Certificate Menu

      Label: cacert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
10 <enter>
```

Figure 45. Creating a Signed Certificate and Key

The **Certificate Usage** menu appears, followed by menus to select the certificate key algorithm and key size (or ECC key type and EC named curve if ECC is selected as the certificate key algorithm. See "Creating a Signed ECC Certificate and Key" on page 273. Once the certificate type is determined, you will be prompted to enter:

- a label to uniquely identify the key and certificate within the key database
- the individual fields within the subject name
- certificate expiration. The valid range for a self-signed certificate is 1 to 9999 days. The default is 365 days.

```
      Certificate Usage

   1 - CA certificate
   2 - User or server certificate

Select certificate usage (press ENTER to return to menu):
2 <enter>

      Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key
   4 - Certificate with a Diffie-Hellman key

Select certificate key algorithm (press ENTER to return to menu):
1 <enter>

      RSA Key Size

   1 - 1024-bit key
   2 - 2048-bit key
   3 - 4096-bit key

Select RSA key size (press ENTER to return to menu):
1 <enter>

Enter label (press ENTER to return to menu):
signedcert <enter>

Enter subject name for certificate
  Common name (required):
My signed Certificate <enter>

  Organizational unit (optional):
ID <enter>
  Organization (required):
IBM <enter>

  City/Locality (optional):
Endicott <enter>

  State/Province (optional):
NY <enter>

  Country/Region (2 characters - required):
US <enter>

Enter number of days certificate will be valid (default 365):
300 <enter>

Enter 1 to specify subject alternate names or 0 to continue:
1

Please wait .....
```

Figure 46. Certificate Type menu

Press option **0** to continue or option **1** to specify the subject alternate names. If option **1** is selected, the **Subject Alternate Name Type** menu appears.

```
 Subject Alternate Name Type

   1 - Directory name (DN)
   2 - Domain name (DNS)
   3 - E-mail address (SMTP)
   4 - Network address (IP)
   5 - Uniform resource identifier (URI)

Select subject alternate name type (press ENTER if name is complete):
1 <enter>

Enter subject name for certificate
  Common name (required):
Test server <enter>

  Organizational unit (optional):
ID <enter>

  Organization (required):
IBM <enter>

  City/Locality (optional):
Endicott <enter>

  State/Province (optional):
NY <enter>

  Country/Region (2 characters - required):
US <enter>

       Subject Alternate Name Type

   1 - Directory name (DN)
   2 - Domain name (DNS)
   3 - E-mail address (SMTP)
   4 - Network address (IP)
   5 - Uniform resource identifier (URI)

Select subject alternate name type (press ENTER if name is complete):
<enter>

Please wait .....
```

Figure 47. Subject Alternate Name menu

When specifying subject alternate names, you are prompted for the type of the alternate name. After the alternate name type is determined, you will be prompted to enter:

- the individual fields within the subject name.

After the individual fields are completed, select option **0** to continue or option **1** to specify another subject alternate name (repeat the process).

Creating a Signed ECC Certificate and Key: If ECC is selected as the certificate key algorithm in the Certificate Key Algorithm menu, you are prompted to choose the ECC key type (for user or server certificates only) to be set in the new certificate and the EC named curve to be used when generating the ECC key.

The following example creates an end-entity certificate with an ECDSA key using a 256-bit NIST suggested named curve.

```
       Certificate Usage

   1 — CA certificate
   2 — User or server certificate

Select certificate usage (press ENTER to return to menu): 2 <enter>

       Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key
   4 - Certificate with a Diffie-Hellman key

Select certificate key algorithm (press ENTER to return to menu): 3 <enter>

       ECC Key Type

   1 — General ECC key
   2 — ECDSA Key
   3 - ECDH key

Select ECC key type (press ENTER to return to menu): 2 <enter>
```

Figure 48. Selecting the ECC Key Type

The selected key type determines the setting of the keyUsage extension in the new certificate. A general ECC key allows Digital Signature, Non-repudiation and Key Agreement. An ECDSA key allows Digital Signature and Non-repudiation. An ECDH key allows Key Agreement only.

If option 1 is selected in the Certificate Usage menu, requesting a CA certificate, the ECC Key Type menu does not appear. The keyUsage extension of the new certificate is set to allow the certificate to be used to sign certificates and certificate revocation lists.

Once the key type has been selected, you are prompted to select the ECC curve type. For a FIPS database, Brainpool standard curves are not supported and, for this reason, the ECC Curve Type menu may not appear.

```
       ECC Curve Type

   1 - NIST recommended curve
   2 - Brainpool standard curve

Select ECC curve type (press ENTER to return to menu): 1 <enter>

       NIST Recommended Curve Type

   1 - secp192r1
   2 - secp224r1
   3 - secp256r1
   4 - secp384r1
   5 - secp521r1

Select NIST recommended curve (press ENTER to return to menu): 3 <enter>

Enter label (press ENTER to return to menu): signedECCcert <enter>
Enter subject name for certificate
  Common name (required): My signed ECC Certificate <enter>
  Organizational unit (optional): ID <enter>
  Organization (required): IBM <enter>
  City/Locality (optional): Endicott <enter>
  State/Province (optional): NY <enter>
  Country/Region (2 characters - required): US <enter>
Enter number of days certificate will be valid (default 365): 300 <enter>

Enter 1 to specify subject alternate names or 0 to continue: 0 <enter>

Please wait .....

Certificate created.

Press ENTER to continue.
```

Figure 49. Selecting the ECC Curve Type

For a FIPS database, some curves may not be recommended for use and may not appear in the ECC Curve Type menu. After selecting the curve type you are prompted to enter the certificate label, subject name, expiration and (optionally) subject alternate names. See"Creating a Signed Certificate and Key" on page 270 for more information.

Creating a certificate to be used with a fixed Diffie-Hellman key exchange: Create a server certificate to be used during an SSL handshake using a fixed Diffie-Hellman key exchange. Fixed Diffie-Hellman requires both sides of the exchange to be based off of the same generation parameters. In order for each side to use the same generation parameters, a key parameter file must be created to be used as input to the certificate being signed.

To create a key parameter file, from the **Database Menu**, enter **6**. You are asked to select the key type and key size. Only 1024-bit DSA keys, 2048-bit DSA keys, or 2048-bit fixed Diffie-Hellman keys are valid for use in a FIPS database. Once the key type is determined, you will be prompted to enter a key parameter file name. The file name is interpreted relative to the current directory when GSKKYMAN is invoked. You may also specify a fully qualified file name.

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

   0 - Exit program

Enter option number:
6 <enter>
```

```
       Key Type

   1 - DSA key
   2 - Diffie-Hellman key

Select key type (press ENTER to return to menu):
2 <enter>

      Diffie-Hellman key

   1 - 1024-bit key
   2 - 2048-bit key

Select Diffie-Hellman key size (press ENTER to return to menu):
1 <enter>

Enter key parameter file name (press ENTER to return to menu):
dh_key_1024.keyfile <enter>

Please wait ......

Key parameter file created.

Press ENTER to continue
```

Figure 50. Creating a key parameter file to be used with Diffie-Hellman

After the key parameter file has been created, the next step is to create the signed certificate using an existing certificate in the key database file to sign the server certificate. From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. In the **Key and Certificate Menu**, choose option **10** to create a signed certificate and key. This requires the displayed certificate to contain an RSA or a DSA key and to have signing capability.

Select the certificate type by choosing either option **9** or **10**, user or server certificate with a Diffie-Hellman key from the **Certificate Type Menu**. The Diffie-Hellman key size must match the key size of the key parameters called previously.

Once the certificate type is determined, you will be prompted to enter:

- key parameter file created earlier
- a label to uniquely identify the key and certificate within the key database
- the individual fields within the subject name
- certificate expiration (Valid expiration range is 1 to 9999 days. Default value is 365 days)
- the subject alternate names (optional).

```
       Certificate Type Menu

   1 - CA certificate
   2 - User or server certificate

Select certificate usage (press ENTER to return to menu):
2 <enter>

       Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key
   4 - Certificate with a Diffie-Hellman key

Select certificate key algorithm (press ENTER to return to menu):
4 <enter>

       Diffie-Hellman Key Size

   1 - 1024-bit key
   2 - 2048-bit key

Select key size (press ENTER to return to menu):
1 <enter>

Enter key parameter file name (press ENTER to return to menu):
dh_key_1024.keyfile <enter>

Enter label (press ENTER to return to menu):
DSA_cert_with_DH_1024_key <enter>

Enter subject name for certificate:
       Common name (required):
DSA cert with DH 1024 key <enter>

       Organizational unit (optional):
Test <enter>

       Organization (required):
Test <enter>

       City/Locality (optional):
Poughkeepsie <enter>

       State/Province (optional):
NY <enter>

       Country/Region (2 characters - required):
US <enter>

Enter number of days certificate will be valid (default 365):
5000 <enter>

Enter 1 to specify subject alternate names or 0 to continue:
 0 <enter>

Please wait .....

Certificate created.

Press ENTER to continue.
```

Figure 51. Creating a certificate to be used with Diffie_Hellman

Once the certificate is created, the next step is to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by SSL without having to specify

its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264.

Creating a Certificate Renewal Request: Certificate renewal requests allow for existing signed certificates that have expired or are nearing their expiration dates to be renewed without having to create a brand new certificate request. The renewed certificate continues to contain the same subject name, public/private key pair. From the **Key Management Menu**, select **1 - Manage keys and certificates** to display the **Key and Certificate Menu**. In the **Key and Certificate Menu**, choose option **11** to create a certificate renewal request.

```
      Key and Certificate Menu

      Label: cacert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
11 <enter>

Enter request filename (press ENTER to return to menu):
renewal.arm <enter>

Certificate request created.

Press ENTER to continue.
```

Figure 52. Select 11 to Create a Certificate Renewal Request

Enter the request file name (press ENTER to return to menu). The certificate request is created. Press enter to continue. After creating the certificate renewal request, perform the following steps:

1. If you want a certificate authority (CA) to sign the certificate, send the certificate request to the CA. See "Sending the Certificate Request" on page 260. If you are acting as your own CA, use the GSKKYMAN command line interface to sign the certificate. See "Using GSKKYMAN to be Your Own Certificate Authority (CA)" on page 282.
2. Receive the renewed certificate into your key database. See "Receiving the Signed Certificate or Renewal Certificate" on page 260.

Importing a Certificate from a File as a Trusted CA Certificate

If you are using a certificate authority for generating your certificates that is not one of the default certificate authorities for which certificates are already stored in the key database, then you must import the certificate authority's certificate into your key database file before you use SSL. If you are using **client authentication**, then the CA certificate must be imported into the key database of the server

program. The client program's key database file must have the CA certificate imported regardless of whether or not the SSL connection uses **client authentication**.

If you are using a self-signed certificate as the SSL server program's certificate and your SSL client program is also using SSL, then you must import the server's self-signed certificate into the client program 's key database file without a private key.

If you are using a self-signed certificate as the SSL client program's certificate and your SSL server program is also using SSL with client authentication requested, then you must import the client's self-signed certificate into the server program's key database file without a private key.

If the CA certificate being imported was signed by another CA certificate, the complete chain must be present in the key database file before the import. If the CA certificates chain consists of more than one certificate and the certificates exist in individual files, you must import the certificates starting with the root CA certificate.

A number of well-known certificate authority's (CA) certificates are stored in the key database when the key database is created. To get a certificate list, select **2 - Manage certificates** from the **Key Management Menu**. The following figures contain lists of CAs for which certificates are stored on key database creation:

```
      Certificate List

      Database: /home/sufwl1/ssl_cmd/mykey.kdb

   1 - VeriSign Class 1 Public Primary CA
   2 - VeriSign Class 2 Public Primary CA
   3 - VeriSign Class 3 Public Primary CA
   4 - Thawte Server CA
   5 - Thawte Premium Server CA
   6 - Thawte Personal Basic CA
   7 - Thawte Personal Freemail CA
   8 - Thawte Personal Premium CA
   9 - Equifax Secure Certificate Authority

   0 - Return to selection menu

Enter label number (ENTER for more labels, p for previous list):
```

Figure 53. Certificate List (part 1)

```
      Certificate List

      Database: /home/sufwl1/ssl_cmd/mykey.kdb

   1 - Equifax Secure eBusiness CA-1
   2 - Equifax Secure eBusiness CA-2
   3 - Equifax Secure Global eBusiness CA-1
   4 - VeriSign Class 1 Public Primary CA - G2
   5 - VeriSign Class 2 Public Primary CA - G2
   6 - VeriSign Class 3 Public Primary CA - G2
   7 - VeriSign Class 4 Public Primary CA - G2
   8 - VeriSign Class 1 Public Primary CA - G3
   9 - VeriSign Class 2 Public Primary CA - G3

   0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
```

Figure 54. Certificate List (part 2)

```
      Certificate List

      Database: /home/RACFU01/mykey.kdb

   1 - VeriSign Class 3 Public Primary CA - G3
   2 - VeriSign Class 4 Public Primary CA - G3
   3 - VeriSign Class 3 Public Primary CA - G5

   0 - Return to selection menu

Enter label number (ENTER for more labels, p for previous list):
```

Figure 55. Certificate List (part 3)

To import a certificate without a private key into your key database file, first get the certificate in a file with the file in either Base64-encoded, Binary encoded or PKCS #7 format. From the **Key Management Menu** enter **7** to import a certificate:

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/mykey.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
7 <enter>

Enter import file name (press ENTER to return to menu):
cert.arm <enter>

Enter label (press ENTER to return to menu):
cacert2 <enter>

Certificate imported.

Press ENTER to continue.
```

Figure 56. Importing a Certificate from a File

You will be prompted to enter the certificate file name and unique label to be assigned to the certificate.

Once the certificate is imported, you will receive a message indicating the import was successful. The certificate is treated as "trusted" so that it can be used in verifying incoming certificates. For a program acting as an SSL server, this certificate is used during the verification of a client's certificate. For a program acting as an SSL client, this certificate is used to verify the server's certificate which is sent to the client during SSL handshake processing.

Importing a Certificate from a File with its Private Key

To store a certificate into a different key database format or to a different system with its private key, the certificate must be exported from the source system into a PKCS #12 format file (See Copying a Certificate with its Private Key for more information). PKCS #12 files are password-protected to allow encryption of the private key information. If the CA certificate being imported was signed by another CA certificate, the complete chain must be present in the key database file prior to the import. From the **Key Management Menu**, enter **8** to import a certificate and a private key:

```
      Key Management Menu

      Database: /home/sufwl1/ssl_cmd/anne.kdb
      Expiration: 2025/12/02 10:11:12

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
8 <enter>

Enter import file name (press ENTER to return to menu):
cert.p12 <enter>

Enter import file password (press ENTER to return to menu):
<enter password>

Enter label (press ENTER to return to menu):
newcert <enter>

Certificate and key imported.

Press ENTER to continue.
```

Figure 57. Importing a Certificate and Private Key from a File

You will be prompted to enter the certificate file name, password and a unique label to be assigned to the certificate.

Once the certificate is imported, you will receive a message indicating that import was successful. The next step is to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by SSL without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 264).

A certificate and key can be imported into a FIPS key database providing it is a PKCS #12 Version 3 with strong encryption format. When adding certificates from the import file to a FIPS key database file only certificates signed with FIPS signature algorithms using FIPS-approved key sizes may be imported. When processing a chain of certificates, processing of the chain will terminate if a non-FIPS certificate is encountered. Certificates processed prior to the failing certificate will be added to the key database file. It is the responsibility of the importer to ensure that the file came from a source meeting FIPS 140-2 criteria in order to maintain adherence to the FIPS criteria.

Using GSKKYMAN to be Your Own Certificate Authority (CA)

The GSKKYMAN command provides the capability for you to act as your own Certificate Authority (CA). Being your own CA allows you to sign your own or anyone else's certificate requests. This is very handy if you only need certificates within your private Web network and not for outside Internet commerce.

To be your own CA in a Web network, you must create a CA database and self-signed CA certificate using GSKKYMAN. A server or client that wishes for you to sign a certificate must supply you with their certificate request. After signing the certificate, the server or client must receive the CA certificate and the newly signed certificate. The CA-signed certificate must then be received into either the client or server key database.

The following table describes the steps needed to become your own CA to allow secure communication between a client and a server. This example reflects the steps followed when the CA is on a different system or is a different user than the issuer of the certificate request.

Certificate Authority (System A)	**Server or Client (System B)**
Step 1 - Create a key database	
Create a key database using the GSKKYMAN command: • From the **Database Menu**, select option **1 - Create new database** See "Creating, Opening and Deleting a Key Database File" on page 248 for details.	Create a key database using the GSKKYMAN command: • From the **Database Menu**, select option **1 - Create new database** See "Creating, Opening and Deleting a Key Database File" on page 248 for details.
Step 2 - Create a Root Certificate Authority certificate	
Create a Certificate Authority certificate: • From the **Key Management Menu**, select option **6 - Create a self-signed certificate** • From the **Certificate Usage Menu,** select option **1 - CA certificate** See "Creating a Self-Signed Server or Client Certificate" on page 253 for details.	No action required.
Step 3 - Create a certificate request	
No action required.	Create a certificate request: • From the **Key Management Menu**, select option **4 - Create new certificate request** See "Creating a Certificate Request and Processing the Signed Request" on page 256 for details.
Step 4 - Send the certificate request to the CA	
No action required.	Send the certificate request to the CA:. See "Sending the Certificate Request" on page 260.
Step 5 - Sign the certificate request	

Certificate Authority (System A)	Server or Client (System B)

Before signing a certificate for a client or server, you need to make sure that the requestor has a legitimate claim to request the certificate. After you have verified the claim, you can create a signed certificate.
 	
To sign the certificate request, the GSKKYMAN command must be issued using command-line options (see "GSKKYMAN Command Line Mode Syntax" on page 285 for a description of the options). The GSKKYMAN command must be issued with the following parameters:
`gskkyman -g -x num-of-valid-days`
`-cr certificate-request-file-name`
`-ct signed-certificate-file-name`
`-k CA-key-database-file-name`
`-l label`
 	
Example: The following command will allow you to sign a request certificate and allow the certificate to be valid for 360 days.
`gskkyman -g -x 360`
`-cr server_request.arm`
`-ct server_signed_cert.arm -k CA.kdb -l labelname`
 	
After you have entered the command, you will be prompted to enter the database password.
Notes:
1. The signed certificate will be an end user certificate unless the -ca option is specified.
2. The filename specified on the -ct option is created for you by the utility, and is the actual signed certificate file.
3. The valid certificate lifetime range is between 1 and 9999 days. The certificate end date will be set to the end date for the CA certificate if the requested certificate lifetime exceeds the CA certificate lifetime.
	No action required.

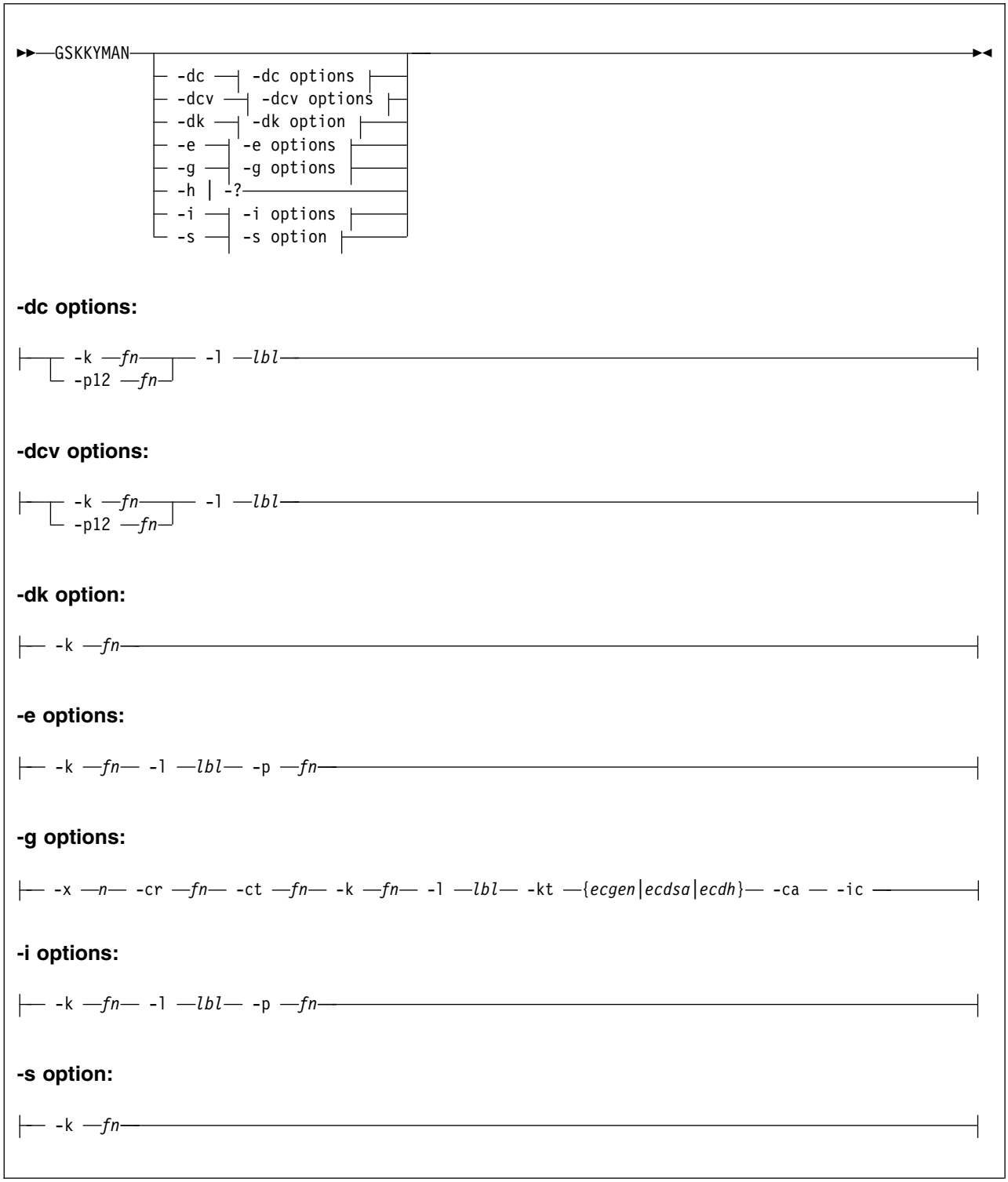
Step 6 - Send the signed CA certificate and the newly signed certificate to the requestor	
Export the signed CA certificate (created in Step 2) to a Base64 file (DER or PKCS #7) See "Copying a Certificate Without its Private Key" on page 265. Send (for example, without its private key ftp) the Base64 file and the newly signed certificate (created in Step 4) to the requestor.	No action required.
Step 7 - Import the CA certificate	
No action required.	Import the CA certificate. See "Importing a Certificate from a File as a Trusted CA Certificate" on page 278.
Step 8 - Receive the signed certificate	
No action required.	Receive the signed certificate. See "Receiving the Signed Certificate or Renewal Certificate" on page 260. **Note:** Depending upon the SSL application, you may need to either send the CA certificate to the client, or the server application may actually present the certificate to the client for them during SSL session setup.

GSKKYMAN Command Line Mode Syntax

This section describes the format and options of the GSKKYMAN command.

GSKKYMAN (gskkyman utility) Command

Format



Purpose

The GSKKYMAN command is used for key database management.

Functions

-dc
Display certificate details

-dcv
Display certificate verbose details

-dk
Display key database expiration and record length

-e Export a certificate and its associated private key

-g Generate signed certificate for a certificate request

-h Display command help

-i Import a certificate and its associated private key

-s Store the database password in the stash file

-? Display command help.

Options

-ca
A certification authority certificate will be generated if **-ca** is specified. An end user certificate will be generated if **-ca** is not specified.

-cr
Specifies the name (*fn*) of the certificate request file. You will be prompted for the file name if this option is not specified.

-ct
Specifies the name (*fn*) of the output generated signed certificate file. You will be prompted for the file name if this option is not specified. You may specify any name. If you specify an existing file name, the file will be overwritten.

-ic
Specifies that the certification chain certificates be included in the certificate file. Otherwise, just the signed certificate will be included in the certificate file.

-k Specifies the name (*fn*) of the key database. This option is mutually exclusive with the **-p12** option. You will be prompted for the key database file name if neither this option nor the **-p12** option is specified. The length of the fully qualified file name cannot exceed 251 characters. If the file name does not end with an extension of 1-3 characters, the length of the fully qualified file name cannot exceed 247 characters. Finally, the key database name cannot end with .rdb or .sth.

-kt
Specifies the key type of the certificate to be created. This option is valid when signing an end user certificate or certificate request containing an ECC public key and affects the settings of the keyUsage extension of the certificate created. Valid key type options are ecgen, ecdsa and ecdh. ecgen creates a certificate with digitalSignature, nonRepudiation and keyAgreement set, ecdsa creates a certificate with digitalSignature and nonRepudiation set, and ecdh creates a certificate with keyAgreement set. If the **-kt** option is not specified for an end

user ECC certificate or certificate request, the default option is ecgen. For other certificate types the **-kt** option is ignored.

-l Specifies the certificate label (*lbl*). The label must be enclosed in double quotation marks if it contains one or more spaces. If the certificate is being used to sign a certificate request (sign function), the certificate must be a CA. The label for the default key will be used if this option is not specified (export or sign function) or you will be prompted for the label (import function).

-p Specifies the name (*fn*) of the PKCS #12 file. You will be prompted for the file name if this option is not specified.

-p12
Specifies the name of the PKCS #12 file containing the certificates to be displayed. It must be a PKCS #12 V3.0 binary format file. This option is mutually exclusive with the **-k** option. The length of the fully qualified file name cannot exceed 251 characters. If the file name does not end with an extension of 1-3 characters, the length of the fully qualified file name cannot exceed 247 characters. Lastly, the PKCS #12 file cannot end with .kdb, .rdb or .sth.

-x Specifies the number of days (*n*) until the signed certificate expires and must be between 1 and 9999 days. The certificate will expire in 365 days if this option is not specified.

Usage

The GSKKYMAN command is used to manage a key database and its associated request database, or to list the contents of a PKCS #12 file. Interactive menus are displayed if no command options are specified. Otherwise, the requested database/PKCS #12 file function is performed and the GSKKYMAN command exits.

Note: The ability to display the contents of a PKCS #12 file is not supported through the interactive menu-driven interface.

If the **-p12** (PKCS #12 file) option is specified with the **-dc** or **-dcv** functions, and the **-l** option is also specified, the certificate with the matching label is displayed. If the **-l** option is not specified, all certificates within the file are displayed.

If the command does not specify the **-p12** option, then it is assumed that the function is to be performed for a key database. If neither the **-k** nor the **-p12** option is specified, the user is prompted for a key database file name.

If both the **-k** and **-p12** options are specified, the command is rejected and an error message is displayed.

The key database contains certificates and private keys and normally has a filename extension of '.kdb'. The request database contains requests for new certificates and always has a filename extension of '.rdb'. The database stash file contains the masked database password and always has a filename extension of '.sth'. Access to these files should be restricted to the database owner.

A certificate or request database consists of fixed-length records. The record length is specified when the database is created and must be large enough to contain the largest certificate entry. A record length of 5000 should be sufficient for most applications. The record length can be increased if necessary after the database has been created.

A temporary database file is created when a database is updated during GSKKYMAN processing. The temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then rewritten and the temporary database file is deleted upon successful completion of the rewrite operation. The temporary database file will not be deleted if an error occurs while rewriting the database file. If this happens, you can replace the database file with the temporary database file in order to recover from the error. If an error does occur and you do not rename or delete the temporary file, you will get an error on the next database update operation indicating the backup file already exists.

If all certificates in a key database are displayed with the -dc or -dcv command, then all certificates with private keys are outputted, followed by all certificates without private keys.

GSKKYMAN Command Line Mode Examples

Command mode is entered when the GSKKYMAN command is entered with parameters. The requested database function will be performed and then the command will exit.

- Store the database password in the stash file

  ```
  gskkyman -s -k filename
  ```

  The database password is masked and written to the key stash file. The file name is the same as the key database file name but has an extension of '.sth'. You will be prompted for the key database file name if the '-k' option is not specified.

- Export a certificate and the associated private key

  ```
  gskkyman -e -k filename -l label -p filename
  ```

  The certificate and associated private key identified by the record label are exported to a file in PKCS #12 Version 3 format using strong encryption. The default key will be exported if the '-l' option is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the export file name if the '-p' option is not specified.

- Import a certificate and associated private key

  ```
  gskkyman -i -k filename -l label -p filename
  ```

  A certificate and associated private key are imported from a file in PKCS #12 format. You will be prompted for the label if the '-l' option is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the import file name if the '-p' option is not specified.

- Create a signed certificate for a certificate request

  ```
  gskkyman -g -x days -cr filename -ct filename -k filename -l label -ca -ic
  ```

  The certificate request identified by the -cr parameter is processed and a signed certificate is created and written to the certificate file identified by the -ct parameter. The -x parameter specifies the number of days until the certificate expires and defaults to 365 days. The certificate is signed using the default key if the -l parameter is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the certificate request file name if the '-cr' option is not specified. You will be prompted for the signed certificate file name if the '-ct' option is not specified.

  The signed certificate will be an end user certificate unless the -ca option is specified. A certification authority certificate will have basic constraints and key usage extensions which allow the certificate to be used to sign other certificates and certificate revocation lists. An end user certificate will have basic constraints and key usage extensions which allow the certificate to be used as follows:

– An RSA key can be used for authentication, digital signature, and data encryption.
– A DSA key can be used for authentication and digital signature.
– An ECC key depends on the key type option supplied. A general ECC key (-kt ecgen) can be used for authentication, digital signature, and key agreement. An ECDSA key (-kt ecdsa) can be used for authentication and digital signature. An ECDH key (-kt ecdh) can be used for key agreement. The default option is ecgen.

Any certificate can be used to sign the new certificate as long as the certificate has a private key, the basic constraints certificate extension (if present) has the CA indicator set, and the key usage certificate extension (if present) allows signing certificates. However, depending upon how the new certificate is subsequently used, it may fail the validation checking if the signing certificate is not a valid certification authority certificate.

The signing algorithm that will be used to sign the new certificate is based on the key algorithm of the signing certificate. An RSA signature will use the most secure and compatible SHA-based hash in use in the signature algorithm of either the signing certificate or the certificate request. A DSA signature will use SHA-1. A DSA signature with a 2048-bit DSA key uses SHA-256. An ECC signature uses the suggested digest for the key size of the ECC private key, as specified in Table 29 on page 236. Possible signing algorithms are:

– x509_alg_sha1WithRsaEncryption
– x509_alg_sha224WithRsaEncryption
– x509_alg_sha256WithRsaEncryption
– x509_alg_sha384WithRsaEncryption
– x509_alg_sha512WithRsaEncryption
– x509_alg_dsaWithSha1.
– x509_alg_dsaWithSha256
– x509_alg_ecdsaWithSha256
– x509_alg_ecdsaWithSha384
– x509_alg_ecdsaWithSha512

The certificate file will contain the generated X.509 certificate in DER-encoded Base64 format if the -ic option is not specified. The certificate file will contain the generated X.509 certificate and the certification chain certificates as a PKCS #7 message in Base64 format if the -ic option is specified.

- Display all certificates in a key database

  ```
  gskkyman -dc -k filename
  ```

  After you are prompted for the key database password, the certificates will be displayed. You are prompted for the key database file name if the -k option is not specified. Because of the number of certificates that can exist in a key database file, it is suggested that you redirect the output to a file. This allows for easy review of the certificates and any post-processing of the certificate output.

- Display key database expiration date

  ```
  gskkyman -dk -k filename
  ```

  After you are prompted for the key database password, the full key database path and file name, expiration date and record length are displayed. You are prompted for the key database file name if the -k option is not specified.

GSKKYMAN Command Line Mode Displays

Command mode is entered when the GSKKYMAN command is entered with parameters. The requested database function will be performed and then the command will exit.

- GSKKYMAN command-mode key database file display

  When the key database password is correctly entered:

  Command:

  ```
  gskkyman -dk -k example.kdb
  ```

  Output:

  ```
  Database: /home/sufwl1/ssl_cmd/example.kdb
  Expiration Date: 2025/12/02 10:11:12
  Record length: 5000
  ```

- GSKKYMAN command-mode certificate display

  Command:

  ```
  gskkyman -dc -k example.kdb -l 'Test User'
  ```

  Output for a single certificate:

  ```
  Label:
          <Test User>
  Trusted:
          Yes
  Version:
          3
  Serial number:
          45ac4d23000a6023
  Issuer's Name:
         <CN=Test User,OU=Test unit,O=IBM, L=Endicott, ST=NY, C=US>
                    Subject's Name:
         <CN=Test User,OU=Test unit,O=IBM, L=Endicott, ST=NY, C=US>
                    Effective date:
         2010/01/16 21:02:02
  Expiration date:
         2010/01/16 21:02:02
  Signature algorithm:
         sha1WithRsaEncryption
  Issuer unique ID:
         None
  Subject unique ID:
         None
  Public key algorithm:
         rsaEncryption
  Public key size:
         1024
  Public key:
         30 81 89 02 81 81 00 9A 9A BC 53 49 50 8B AF F9
         AF 00 A1 F3 A6 80 3A DA 2C A5 7C 65 A0 00 96 FA
         1A 71 74 74 B4 2A 95 92 AC 1D 76 F1 97 37 D3 BC
         06 8B DC 83 2F 7F 08 B0 EA 1F F8 71 AC 8F 96 3E
         6E DA F5 F8 D0 A6 51 A4 AF E6 21 F5 50 AC B7 06
         83 BF 88 48 DF 51 DB 18 BF EC 7C 72 DA ED 6C 82
         28 93 7C AE 12 E8 CD 55 16 E1 05 53 63 C1 84 D1
         91 AD 3E E5 70 87 00 0C 14 40 92 D9 6E DD ED 07
         81 9D 93 34 DC 1F 05 02 03 01 00 01
  Private key:
         Yes
  Default key:
         No
  Certificate extensions:
         4
  ```

- GSKKYMAN command-mode certificate display (verbose)

  Command:

  ```
  gskkyman -dcv -k example.kdb -l 'Test User'
  ```

Verbose output for a single certificate:

```
Label:
        <Test User>
Trusted:
        Yes
Version:
        3
Serial number:
        45ac4d23000a6023
Issuer's Name:
       <CN=Test CA,OU=Test unit,O=IBM, L=Endicott, ST=NY, C=US>
Subject's Name:
       <CN=Test User,OU=Test unit,O=IBM, L=Endicott, ST=NY, C=US>
Effective date:
       2010/01/16 21:02:02
Expiration date:
       2010/01/16 21:02:02
Signature algorithm:
       sha1WithRsaEncryption
Issuer unique ID:
       None
Subject unique ID:
       None
Public key algorithm:
       rsaEncryption
Public key size:
       1024
Public key:
       30 81 89 02 81 81 00 9A 9A BC 53 49 50 8B AF F9
       AF 00 A1 F3 A6 80 3A DA 2C A5 7C 65 A0 00 96 FA
       1A 71 74 74 B4 2A 95 92 AC 1D 76 F1 97 37 D3 BC
       06 8B DC 83 2F 7F 08 B0 EA 1F F8 71 AC 8F 96 3E
       6E DA F5 F8 D0 A6 51 A4 AF E6 21 F5 50 AC B7 06
       83 BF 88 48 DF 51 DB 18 BF EC 7C 72 DA ED 6C 82
       28 93 7C AE 12 E8 CD 55 16 E1 05 53 63 C1 84 D1
       91 AD 3E E5 70 87 00 0C 14 40 92 D9 6E DD ED 07
       81 9D 93 34 DC 1F 05 02 03 01 00 01
Private key:
       Yes
Default key:
       No
Critical Extension:
       keyUsage:
               Digital signature
               Non-repudiation
               Key encipherment
               Data encipherment
Non-critical Extension: 1
       subjectAltName:
               EMAIL:
                       <test@ibm.com>
Non-critical Extension: 2
       subjectKeyIdentifier:
               91 DA 60 24 00 31 0A 75 39 F4 F6 56 D5 AD 35 35
               86 2D C6 F8
Non-critical Extension: 3
        authorityKeyIdentifier:
                Key ID:
                     19 6E 03 37 AB 8B 0F 7B 9D A3 A6 8F CC B4 A2 CA
                     AC FA B6 E8
```

Certificate (Key) Database Administration

The key database contains the public-keys that are associated with signers of certificates. These public-keys are, in reality, contained in certificates themselves. Thus, verifying one certificate requires the use of a different certificate, the signer's certificate. In this fashion, a chain of certificates is established, with one certificate being verified by using another certificate and that certificate being verified by yet another certificate, and so on. A certificate, and its associated public key, can be

defined as a *root* certificate. A root certificate is *self-signed*, meaning that the public-key contained in the certificate is used to sign the certificate. Using a root certificate implies that the user *trusts* the root certificate.

The key database used by the SSL server must contain a sufficient number of certificates to allow for verification of the certificates provided by remote hosts during the start up of an SSL/TLS connection. If *either* certificate is self-signed, then that certificate must be stored in each host's key database. If the certificates are signed by some other certificate signer, then the signer's certificate and any certificates that this certificate depends upon must be stored in the key databases. The key databases used by the SSL server must also contain the certificates that it will transmit to a peer host during the startup of an SSL/TLS-protected communication.

Creating and Using a Key Database

The SSL server requires that a key database be set up before SSL/TLS-protected communications can be established. The key database, which contains PKI private keys and certificates, is a password protected file stored in the Byte File System. This file is created and managed using a utility program called **gskkyman**. For detailed information about using the **gskkyman** utility, see "SSL Certificate Management" on page 233.

The key database file that is created must be accessible by the SSL server. The default environment established through installation of the z/VM deliverable provides for such access, with BFS file space (GSKSSLDB) already allocated for maintenance of a key database. This filespace is owned by an IBM-defined user ID (**GSKADMIN**) that serves as a designated key database administrative user ID.

z/VM Certificate Label Requirements
The labels for certificates that are to be used by the SSL server (whether those certificates are server certificates or self-signed certificates) must be no more than eight characters, and must be comprised of only upper case, alphanumeric characters.

For testing purposes, the SSL server can use a self-signed certificate as a server certificate. In this case, the self-signed certificate used by the SSL server must also be stored in the key database referenced by a peer client or server, in order to establish SSL/TLS-protected communications.

Obtaining a Certificate

SSL server certificates can be obtained by contacting a certificate authority (CA) and requesting a certificate. Utilities to formulate a certificate request are provided by **gskkyman**. This certificate request is usually passed to the CA by means of an electronic mail message or by an HTML form which is filled out using a web browser. Once the CA verifies the information for the SSL server, a certificate is returned to the requester, usually by an electronic mail message. The contents of the mail message are used to define the certificate in the key database.

For detailed information about storing CA certificates in the key database using the **gskkyman** utility, see "SSL Certificate Management" on page 233.

Using Self-Signed Certificates

For testing purposes, the SSL server can use a self-signed certificate as a server certificate. To establish SSL/TLS-protected communications using such a certificate, the self-signed certificate must exist in the local key database of each connection participant.

A self-signed certificate, presented by a remote peer, cannot be used unless it has been added to the SSL server certificate database. Such a certificate cannot be used if it has expired, or is not yet valid. As with certificates that are not self-signed, other conditions, such as an invalid signature or incorrect name, can preclude the use of a self-signed certificate.

Note: A self-signed end-entity certificate (server or client certificate) is not recommended for use in production environments and should only be used to facilitate test environments prior to production. Self-signed certificates do not imply any level of security or authenticity of the certificate because, as their name implies, they are signed by the same key that is contained in the certificate. On the other hand, certificates that are signed by a certificate authority indicate that, at least at the time of signature, the certificate authority approved the information contained in the certificate.

Certificate Management

Because the key database is managed independent of the SSL server through use of the **gskkyman** utility, changes to the database, such as the addition or removal of a certificate or alteration of a certificate label, are not automatically reflected in the SSL server (while that server is in operation). Thus, existing secure connections are not affected by these types of changes. For an active SSL server to employ key database content changes when new connections are processed, the **SSLADMIN REFRESH** command must be issued once those changes are complete.

SSL Tracing Information

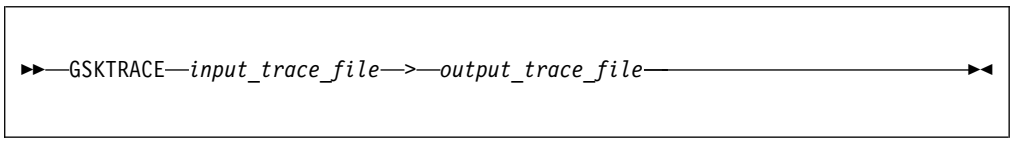
The information and techniques described in this topic are for use primarily by IBM service personnel in determining the cause of an SSL problem. If you encounter a problem and call the IBM Support Center, you may be asked to obtain trace information or enable one or more of the diagnostic messages described below.

The facilities described below are not intended for use in a production environment and are for diagnostic purposes only.

Use the **gsktrace** utility to create a readable copy of SSL trace information. Run **gsktrace** from CMS with the GSKTRACE command. You need access to the TCPMAINT 591 disk where the GSKTRACE EXEC, load module, and message catalogs reside. By default, **gsktrace** accesses the US English message catalogs. If you want **gsktrace** to access the Japanese message catalogs instead, set the NLSPATH environment variable by executing the following commands from a REXX EXEC:

```
nlspath = '/usr/lib/nls/msg/Ja_JP.IBM-939/%N'
'globalv select cenv put nlspath'
```

GSKTRACE (gsktrace utility) command



Purpose

The GSKTRACE command creates a readable copy of SSL trace information.

Note: In order to capture trace information, the trace environment variables **GSK_TRACE** and **GSK_TRACE_FILE** must be set prior to starting the server. A single trace file is created, and there is no limit on the size of the trace file. See "Usage Note."

Operands

input_trace_file
: Specifies the file designated by the GSK_TRACE_FILE environment variable. See "Usage Note."

output_trace_file
: Specifies the name of the output file that will contain the formatted trace information.

Usage Note

In order to capture trace information, you must set the **GSK_TRACE** and **GSK_TRACE_FILE** prior to starting the server. Specify these environment variables in the CENV group of GLOBALV.

- **GSK_TRACE**

  Specifies a bit mask enabling SSL trace options. No trace option is enabled if the bit mask is 0 and all trace options are enabled if the bit mask is 0xffff. The bit mask can be specified as a decimal (nnn), octal (0nnnn) or hexadecimal (0xhh) value.

  The following trace options are available:

  0x01 = Trace function entry
  0x02 = Trace function exit
  0x04 = Trace errors
  0x08 = Include informational messages
  0x10 = Include EBCDIC data dumps
  0x20 = Include ASCII data dumps

  Note: The trace mask can also be set by the GSKTRACE operand on the SSL server VMSSL command. See *z/VM: TCP/IP Planning and Customization*.

- **GSK_TRACE_FILE**

  Specifies the name of the trace file and defaults to /tmp/gskssl.%.trc. The trace file is not used if the **GSK_TRACE** environment variable is not defined or is set to 0.

The current process identifier is included as part of the trace file name when the name contains a percent sign (%). For example, if GSK_TRACE_FILE is set to `/tmp/gskssl.%.trc` and the current process identifier is 247, then the trace file name will be `/tmp/gskssl.247.trc`.

Note: It is recommended that if the default trace file value is not being used, the trace file name always contain a '%' character.

Chapter 9. Using the Network File System Commands

NFS is a TCP/IP protocol that allows heterogeneous systems to share data and files across networks. The NFS protocol is a client/server protocol — an NFS *server* runs on a system that has data and files to share, while an NFS *client* runs on a system where data and files are to be shared from NFS servers.

The VM NFS client allows BFS users and applications transparent access to data on remote systems that have NFS servers supporting the SUN NFS Version 2 or Version 3 protocols, or both. These NFS protocols are described in RFCs 1094 and 1813 respectively. z/OS OS/390, Windows XP, Windows 2000, Windows 95, Windows NT, AIX®, LINUX, UNIX systems, and z/VM are examples of these remote systems.

For VM NFS client information, see the OPENVM MOUNT command in the *z/VM: OpenExtensions Commands Reference*.

VM's NFS Server Support

The following NFS information is described in this chapter:

- Sample remote NFS client commands and PCNFSD User ID Authentication
- Diagnosing Mount Problems
- Errors Using Mounted Systems
- Notes for BFS files and directories
- Notes for SFS files and directories
- Notes for minidisk files
- SMSG VM NFS server commands
- Name translation file
- NFS Client Problems
- Deleting CMS record-length fields
- Using NFS with RACF

BFS All functions defined by the NFS protocol are supported by BFS file systems. For information on special considerations, see "Notes for BFS Files and Directories" on page 313.

SFS Most functions defined by the NFS protocol are supported by SFS file systems. For information about which functions are not supported, see "Notes for SFS Files and Directories" on page 314.

Minidisk
Some functions that are defined by the NFS protocol are not supported by CMS minidisk file systems. Others are partially supported. For information about which functions are not supported, see "Notes for Minidisk Files" on page 316.

Network File System (NFS) server support for z/VM is available at the Version 2 level (RFC 1094) and at the Version 3 level (RFC 1813). Both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) transport protocols are supported. If your NFS client can be configured to use NFS Version 3 or the TCP transport protocol, you will be able to select these options.

© Copyright IBM Corp. 1987, 2016

NFS Client Commands

The actual format and use of NFS commands in a client system depends on the implementation of the NFS client. In the following descriptions, information is provided for data that is transmitted by the client system to the VM NFS server. For information about other parts of these commands or data that is interpreted by the client system, consult the documentation for the specific NFS client system used.

You can use the following NFS commands from an NFS client machine.

MOUNT
Mounts a BFS directory, SFS directory, or CMS minidisk on a local directory.

MOUNTPW
Sends authentication information (user IDs and passwords) to the VM NFS server to obtain access to protected CMS files, directories, and file systems.

UMOUNT
Removes mounted CMS file systems in the client environment.

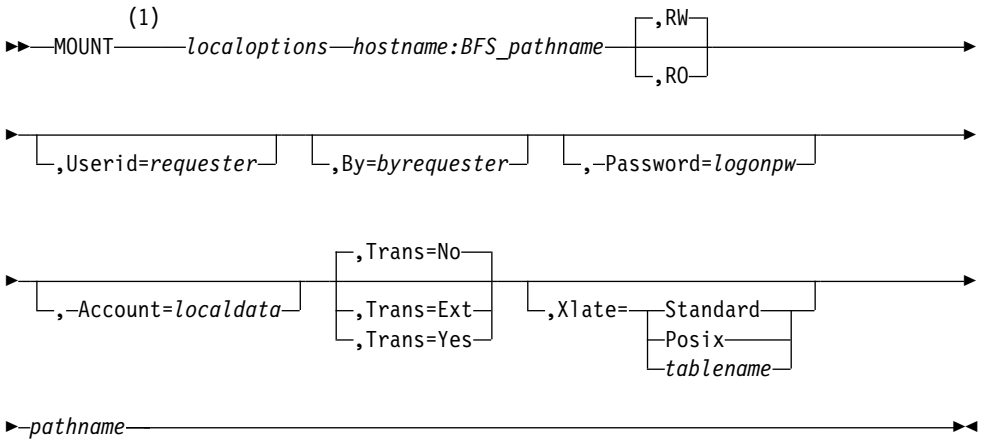
No unique error messages are associated with these commands. The error messages that result from a failed mount attempt depend on the client system, which should provide you with sufficient status information to understand why the failure occurred.

The NFS commands are described in the following sections. In some environments, the command syntax is case-sensitive. The following syntax diagrams are examples only.

MOUNT Command

BFS MOUNT Command Syntax

BFS file system protocols are similar to those of NFS.

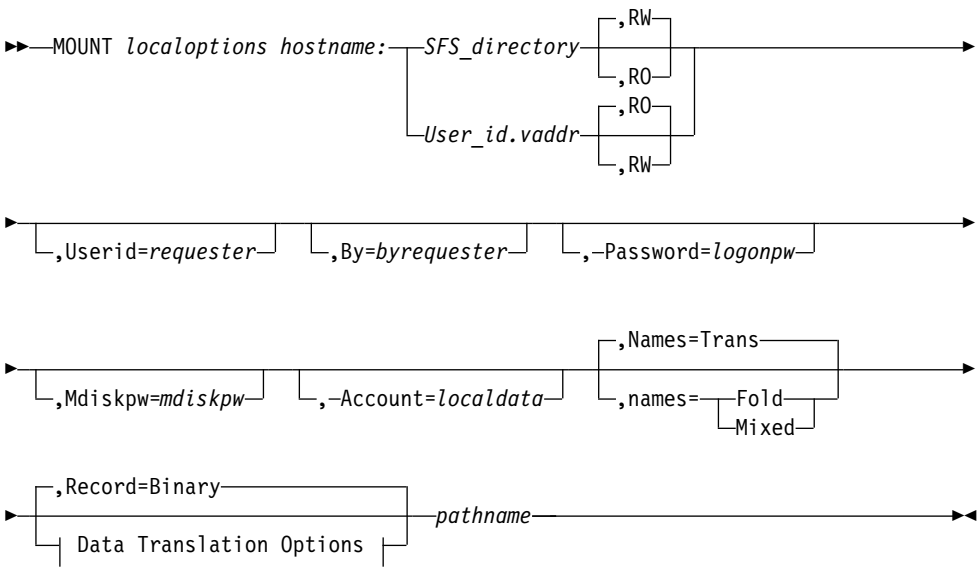


Notes:

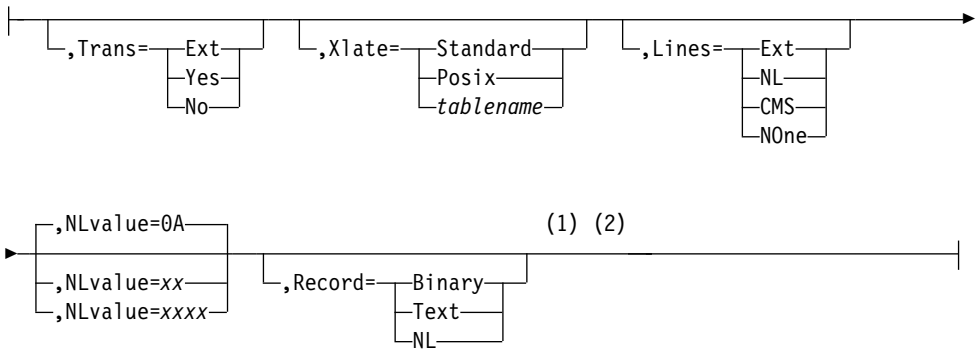
1 *names=*, *nlvalue=* and *record=* are not used if specified when mounting a BFS directory. Keyword options and values that are uppercase in the syntax diagram are the minimum abbreviation for such keyword options or values.

SFS and Minidisk MOUNT Command Syntax

SFS and minidisk file system protocols are very different from those of NFS in both structure and naming convention. SFS and minidisks files have records; the **lines** and **nlvalue** options allow you to tell NFS how to map records into a data stream. The **names** option tells NFS how to map a CMS file identifier (FILENAME FILETYPE) into the NFS file naming convention.



Data Translation Options:



Notes:

1 The use of *trans*= and *lines*= together with *record*= may result in overriding options.

2 *record=binary* is the default only if neither *lines* nor *trans* is specified. Keyword options and values that are uppercase in the syntax diagram are the minimum abbreviation for such keyword options or values.

Purpose

To access a CMS file system, issue the MOUNT command on the NFS client machine in the following format:

MOUNT and MOUNTPW Operands

The specific MOUNT parameters used by an NFS client can vary for different VM systems, based on the access control scheme in use (for example, protecting minidisks using CP LINK passwords versus an external security manager (ESM), such as RACF).

Ensure the appropriate authorizations are in place before issuing a mount request. For example, in the case where an ESM such as RACF is in place to protect minidisks, the owner of a minidisk to be mounted may need to issue RACFPERM or RACFBAT commands to authorize your user ID and/or the VM NFS server to obtain access to that disk. Also, ensure that both the **password=** and **userid=** parameters are included in the minidisk mount request.

For BFS and SFS files, authorization checking is done in the file pool server machine when you request access to files in the mounted directory. Your requests for file access are authorized (or not authorized) based on how you are identified to the file pool server. The VM NFS server uses the verified **userid=** parameter to identify your requests.

Optional parameters can be specified in any order. If an optional parameter is specified more than once, or if overriding parameters such as **record=** and **trans=** are used together, the last instance prevails.

localoptions
: Specifies the MOUNT options, which vary between different client systems. Some common options for a UNIX client are: intr, soft, retry=n, retrans=n, and timeo=n. When a time-out value is specified, it usually is in units of one-tenth of a second; therefore `timeo=20` denotes a time-out value of 2 seconds before a client assumes a network error occurred and retransmits a request to a server.

hostname:
: The *hostname* is the name of the host on which the VM NFS server resides. Some NFS clients use a different syntax to specify the server location in a MOUNT command, rather than placing it as a prefix to the characters that identify the remote file system to be mounted.

BFS_pathname
: Specifies the fully-qualified name of the BFS directory that contains the files and subdirectories to be accessed by the client.

  Use two consecutive commas (**,,**) to indicate that a BFS directory name contains a comma.

  Some NFS clients have difficulties with the special characters required for fully-qualified BFS directory names. You may use slashes (/) in place of the colons (:) that are required before file pool ID and file space ID in a fully-qualified BFS directory name. NFS clients may have other requirements for entering pathnames containing special characters such as blanks.

SFS_directory
: Specifies the SFS file pool directory that contains the files and subdirectories to be accessed by the client. *SFS_directory* must be fully qualified, except that the file space portion of the directory name can default to *requester*.

  Some NFS clients have difficulties with the special characters required for fully-qualified SFS directory names. You may enter a slash (/) in place of the colon (:) that is required between file pool ID and file space ID in a fully-qualified SFS directory name. You may also enter a slash in place of the period (.) that separates subdirectories.

user_id.vaddr
: Specifies the minidisk. The *user_id.vaddr* string defines the minidisk that contains the files to be accessed by the client.

RW Specifies that the type of mount is to be read/write access. This is the default for the mount of a BFS or SFS directory.

RO Specifies that the type of mount is to be read-only. This is the default for a mount of a minidisk.

Userid=*requester*
Specifies a CP (VM) *requester* user ID to be associated with the mount request.

The user ID is used as access control for SFS files, and the user ID is translated into UID and GID values which are used as access controls for BFS files. (Additional access controls may be used if an external security manager (ESM) is active for the file pool.) *Requester* should be enrolled in the file pool containing the BFS or SFS directory, or PUBLIC should be enrolled.

User ID may be used by an external security manager, such as RACF, for deciding whether an NFS client is permitted to access an ESM-protected minidisk.

userid= need not be specified on the MOUNT or MOUNTPW requests if PCNFSD is used.

If **by=** is not specified, the combination of **userid=***requester* and **password=***logonpw* is used by the VM NFS server to validate the identity of the client.

When **userid=** and **by=** are not provided by PCNFSD, MOUNTPW, or a MOUNT request, the mount will be successful only if anonymous mounts are allowed by the VM NFS server. If anonymous MOUNTs are allowed, the mount may be successful if a user ID of ANONYMOU has authority to use the SFS directory or ESM-protected minidisk. For BFS directories, the mount may be successful if the default user UID or GID values allow permission to use the directory. See *z/VM: TCP/IP Planning and Customization* for information about allowing anonymous MOUNTs.

By=*byrequester*
If both **userid=***requester* and **by=***byrequester* are specified, the VM NFS server verifies that **password=***logonpw* is the correct logon password for *byrequester*, and that *byrequester* has LOGON BY privileges for **userid=***requester*.

Password=*logonpw*
Specifies the VM logon password of the user ID requesting the mount.

password= need not be specified on the MOUNT or MOUNTPW requests if PCNFSD is used.

The **-v** option on the OS/2 MOUNT command can be used to have the client prompt you for the password, unless you are mounting a BFS directory.

Mdiskpw=*mdiskpw*
When no external security manger (ESM) is in use, specifies a CP link password when a password is required to access the minidisk being mounted. If a CMS disk is public (that is, its CP LINK password is ALL), no password needs to be provided with the MOUNT command in order to access files on that disk.

The VM NFS server will not issue a CP LINK for an MW link. If a mount request specifies write access for a minidisk, but a write link to that minidisk already exists, the VM NFS server returns a status code to the NFS client which indicates the CMS minidisk is a read-only file system.

Note: If **userid=**, **by=** and **mdiskpw=** are not specified, the password specified in the **password=***logonpw* parameter is used as a minidisk link password for CMS minidisks. This exception is made because earlier releases of VM NFS did not make a distinction between logon and minidisk link passwords.

Account=*local_data*
Allows additional information to be passed to software such as an ESM in the host. Specifies up to 64 characters of information that can be used by customer-supplied programming in the VM NFS server to further control or record access by NFS clients. The format of this data depends upon the supporting software. If account information is specified by a client and there is no supporting software in the server, the account data is ignored by the VM NFS server.

Names=
Specifies the handling of file names for minidisk, SFS files and directories. The names options are:

Fold File names supplied by the client are translated to uppercase; file names supplied to the client are translated to lowercase.

Mixed File names are not changed. The client must use valid CMS names. This mode must be used to refer to CMS files that contain lowercase letters in their name.

Trans Use default name handling, which is described in the following paragraph.

If the names parameter is omitted, the default provides translation for names that are too long or contain invalid characters. File names supplied by the client are translated to uppercase; file names supplied to the client are translated to lowercase. Correct CMS file names are generated by the VM NFS server, and these names are used rather than the actual client names. The VM NFS server creates the translation file ##NFS## #NAMES# on the CMS minidisk for minidisk files, and in the top SFS directory of the file space for SFS files. This translation file contains both the original client file names and the corresponding CMS file names. When file names are read by a client, the inverse translation is performed so that the client sees the original name specified when the file was created.

Trans=
defines whether translation should occur for data in files.

Ext EBCDIC-ASCII translation is done based on the value of the file extension. See Table 31 on page 305.

Yes EBCDIC-ASCII translation is performed.

No No translation is performed.

If neither **record** or **trans** is specified, **trans=no** is the default.

NFS uses translation tables to convert transmitted data between EBCDIC and ASCII. These translation tables can be customized by your TCP/IP administrator. Contact your TCP/IP administrator for information about data translation.

Xlate=
Defines which translation table is to be used for file data translation.

Standard
TCP/IP's standard translation table is to be used.

Posix This table translates ASCII (ISO 8859-1) to and from EBCDIC (IBM-1047). The UNIX line terminator (lf - X'0A') is translated to the VM OpenExtensions line-end character NL - X'15').

tablename
The name of the translate table to be used. Some examples of *tablename* include "Xlate=UK," "Xlate=French," or "Xlate=10471252."

If **xlate** is not specified, a system-defined translation table is used.

tablename may not be abbreviated.

Your TCP/IP administrator may change the list of tables provided, or customize the translation tables in the list. Contact your TCP/IP administrator for information about data translation.

Lines=
defines the way CMS records are converted to an NFS data stream.

Ext The type of conversion done is based on the value of the file extension. See Table 31 on page 305.

NL Line feed characters are inserted at CMS record boundaries.

CMS The CMS file format is maintained. When dealing with CMS variable-length record files, the binary, unsigned, two-byte length field is visible to the client at the beginning of CMS records read from the VM NFS server, and this field must be supplied by the client program in data written to VM.

NOne No linefeed characters are inserted. When dealing with CMS variable-length record files, the two-byte length fields are not visible.

If neither **record** nor **lines** is specified, **lines=none** is the default.

NLvalue=
Specifies two or four hexadecimal characters to be used as the boundary indicating the end of a CMS record. The default is the line feed character.

For example, you can specify *nlvalue=0D0A* for carriage return and line feed.

Record=
The **record** option is supported for compatibility. Use the **trans** and **lines** options to tell NFS how to translate file data.
- **Binary** is equivalent to **trans=no** and **lines=CMS**.
- **Text** is equivalent to **trans=yes** and **lines=CMS**.
- **NL** is equivalent to **trans=yes** and **lines=NL**.

pathname
Specifies the local path name, which identifies where to mount CMS files in the client file name hierarchy.

Figure 58 on page 305 illustrates the MOUNT command.

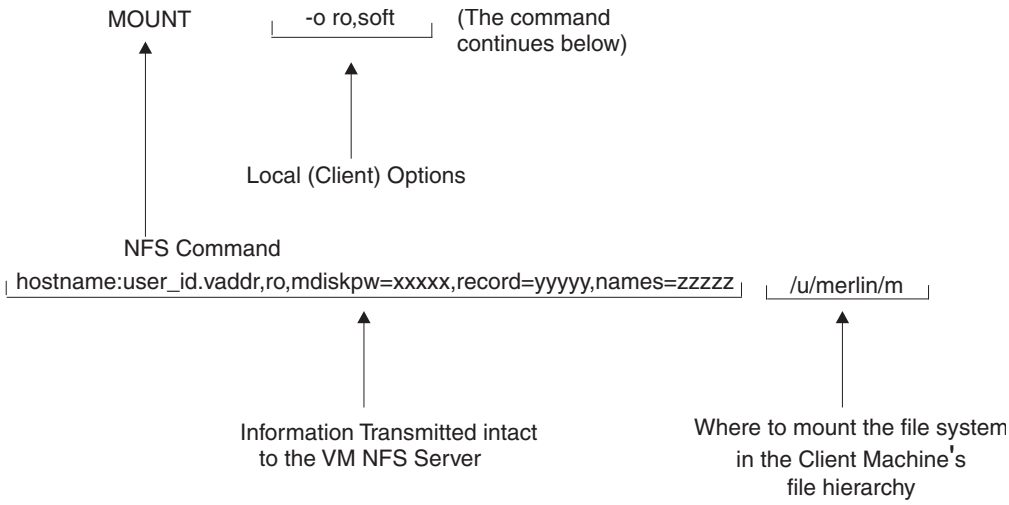


Figure 58. NFS MOUNT Command

NFS File Extension Defaults

Following are the default values used when the **trans=ext** or **lines=ext** options are used. The system administrator for your system may change or add to these default values.

File extension is defined to be the last component of a file name, that is, the characters following the last period in the path name. Up to eight characters are matched, and mixed case is not respected.

Table 31. File Extension Translation Table

File Extension					trans=ext	lines=ext
*BIN[1]					no	none
$EXEC	$REXX	$XEDIT	AMS	AMSERV	yes	NL
ANN	ANNOUNCE	APP	APPEND	ASC		
ASCII	ASM	ASM3705	ASSEMBLE	AVL		
AVAIL	A37	BASDATA	BASIC	BKS		
BKSHELF	C	C++	CAT	CATALOG		
CNTRL	COB	COBOL	COPY	CPP		
DIRECT	DLCS	DOCUMENT	ESERV	EXC		
EXEC	FFT	FOR	FORM	FORTRAN		
FREEFORT	GCS	GROUP	H	HPP		
HTM	HTML	H++	JOB	LISTING		

Table 31. File Extension Translation Table (continued)

File Extension					trans=ext	lines=ext
LOG	LST	MAC	MACLIB	MACRO	yes	NL
MAK	MAKE	ME	MEMBER	MEMO		
MODULE	NAM	NAMES	NETLOG	NONE		
NOT	NOTE	NOTEBOOK	OFS*[2]	OPT		
OPTIONS	PACKAGE	PASCAL	PKG	PLAS		
PLI	PLIOPT	PLS	PVT	REXX		
RPG	SCR	SCRIPT	STY	STYLE		
TEXT	TEXTXXXX	TXT	TXTXXXX	UPDATE		
UPDT	VMT	VSBASIC	VSBDATA	XED		
XEDIT						
other or *none*					no	none

Note:

1. *BIN represents a wildcard, that is, any file extension ending in 'BIN'.
2. OFS* represents a wildcard, that is, any file extension starting with 'OFS' unless it also ends in 'BIN'.

PCNFSD User ID Authentication

Many NFS client implementations contain calls to PCNFSD user ID Authentication services. The purpose of these calls is to present a user ID and password to be verified by the host system. Once verified, the UID and GID by which that user is known at the host is returned to the client. For VM, the POSIXINFO and POSIXGLIST CP directory entry statements define the UID and GID(s) associated with a user ID.

Note that a value of -1 is returned for a user ID of ANONYMOU.

When PCNFSD is used, there is no need to send user ID and logon password in the MOUNT or MOUNTPW. If user ID or password are not provided by MOUNTPW or MOUNT, the PCNFSD values are used by the NFS server, as long as the MOUNT request is received by the NFS server immediately following the PCNFSD request. (The NFS client typically sends the MOUNT request immediately following the PCNFSD information.)

Your system administrator has the option of disabling PCNFSD for your NFS server. Contact your system administrator if you receive a message that PCNFSD is not running or is inaccessible.

NFS MOUNT Examples

In the following example, a MOUNT command is issued from an OS/2 command prompt to mount a BFS directory that resides in the *root* file system in file pool: *fpcool*. Note that VMBFS, FPCOOL and ROOT must be uppercase when specified in the pathname.

1. Mounting a BFS directory:

```
mount -u4189 -g513 x: gd3vm0:/../VMBFS:FPCOOL:ROOT/u/jake,
     userid=elwood,password=mypass
```

In the previous command, the user specified his VM uid and gid via the -u and -v options. This indicates to OS/2 what uid and gid are being used. However, the VM NFS server will perform authorization checking based on the verified **userid=** parameter.

Alternatively, Elwood's VM logon password can be provided via a previous MOUNTPW command.

2. Mounting an SFS directory:

```
mount -v x: gd3vm0:fpcool:jake.mission,trans=ext,lines=ext,
     userid=elwood
```

In the above example, a MOUNT command is issued from an OS/2 command prompt to mount the *mission* SFS subdirectory owned by the VM user ID, *JAKE* which resides in file pool *fpcool*:

After issuing the command above, the user will be prompted for Elwood's user ID password. In response to this prompt, the VM logon (or sign-on) password should be provided.

Alternatively, Elwood's VM logon password can be provided via the **password=** parameter and no password prompt is issued.

In the following examples, MOUNT commands are issued from an OS/2 command prompt to mount the VM 191 minidisk owned by the VM user ID, *JAKE*:

1. Accessing a VM minidisk when an External Security Manager (ESM) is in use:

```
mount -v x: gd3vm0:jake.191,ro,trans=yes,lines=NL,
     userid=elwood
```

After issuing the command above, the user will be prompted for Elwood's user ID password. In response to this prompt, the VM logon (or sign-on) password should be provided.

```
mount x: gd3vm0:jake.191,ro,trans=yes,lines=NL,password=harpman,
     userid=elwood
```

In the above command, Elwood's VM logon password *harpman* is provided via the **password=** parameter; no password prompt is issued.

2. Accessing a VM minidisk when no External Security Manager (ESM) is in use:

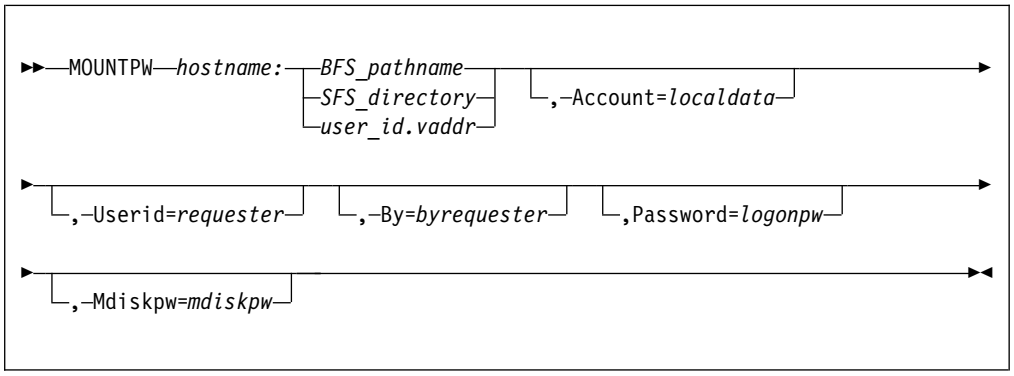
```
mount -v x: gd3vm0:elwood.191,ro,trans=yes,lines=NL
```

After issuing the above command, the user will be prompted for a password. In response to this prompt, a VM minidisk (CP LINK) password should be provided. In this example, read-only access was requested, so the minidisk "Read (R)" password should be provided.

```
mount x: gd3vm0:elwood.191,rw,trans=yes,lines=NL,mdiskpw=whtoast
```

In the above example, read/write access was requested, so the minidisk "Mult (M)" password ("whtoast") was provided via the **mdisk=** parameter.

MOUNTPW Command



Purpose

Use the MOUNTPW command to send authentication information to the VM NFS server to obtain access to protected CMS files, directories and file systems.

Note: If your NFS client is configured to call PCNFSD user ID Authentication services and PCNFSD is available on your VM NFS server, using MOUNTPW is not necessary, because user ID and password information is passed on PCNFSD requests. This applies to mounts for SFS directories, BFS directories, and minidisks protected by ESMs. A minidisk protected by link passwords must still provide the password on the MOUNT or MOUNTPW command.

The MOUNT command can pose a security problem, because NFS client systems often store this command's argument values to respond to a later query asking for information about the mounts that are currently in effect. A password supplied in an argument to a MOUNT command could then be revealed in the query response to someone other than the user who executed the MOUNT command. The MOUNTPW command provides an alternative path for sending passwords, account, and user ID information to the VM NFS server. This information can then be omitted from the subsequent related MOUNT command, and therefore is not present in a display of currently mounted file systems.

A MOUNTPW command precedes the related MOUNT command, which must follow within 5 minutes. After 5 minutes, the VM NFS server discards the information it received in a MOUNTPW command. If a second MOUNTPW command is issued for the same CMS disk or directory before a MOUNT command for that object, the data from the first MOUNTPW command is discarded.

The only MOUNT command options that are recognized on a MOUNTPW command are **userid**, **by**, **password**, **mdiskpw** and **account**. All other MOUNT command options are ignored. If user IDs, passwords, or account value are specified on both a MOUNTPW command and a related MOUNT command, the value from the MOUNT command is used.

MOUNT and MOUNTPW Operands

Refer to the "MOUNT Command" on page 298 for a description of the operands.

Usage Note

The MOUNTPW C source file is supplied on TCPMAINT's 592 disk, as are the executable modules built for the IBM client environments listed in Table 32

Table 32. Executables

Operating Environment	MOUNTPW Executable File
AIX on an IBM RISC System/6000	6000@BIN MOUNTPW
OS/2	OS2@BIN MOUNTPW

If none of the executable modules are appropriate, then the MOUNTPW C file must be copied to the client system and compiled into an executable program before you can execute the MOUNTPW command.

Compile the command using a C compiler resident on the client system. The location of header files (.h) in the MOUNTPW source program may not be where your client system expects for a compile. If this is the case, modify the MOUNTPW source program to specify the correct location for your client system. Some platforms may require that you specify libraries to build the MOUNTPW executable. For example, DYNIX/ptx® requires the following: –lsocket, –lnsl, and –lrpc.

Binary files are supplied containing executable MOUNTPW modules for the IBM AIX environment. These can be used in the NFS client environment AIX Version 4.2.1 and above as an alternative to building executable modules from the source files. Use FTP to copy the files to the AIX environment, renaming the files to the command name and making sure to use the FTP command `binary` to ensure that a binary copy is performed. Once the files are copied to the AIX environment, use the command `chmod a+x mountpw` to make the files executable.

UMOUNT Command

Purpose

Use the UMOUNT command to unmount a currently mounted file system.

The format of the UMOUNT command, like the MOUNT command, depends on the client system.

The most common format of the UMOUNT command is shown above.

Operands

pathname
Specifies the local path name that identifies a mounted remote file system in the client environment.

Figure 59 illustrates the NFS UMOUNT command.

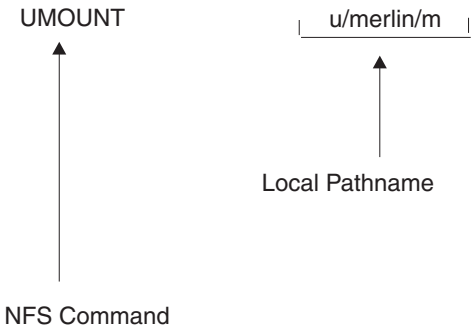


Figure 59. NFS UMOUNT Command

Diagnosing Mount Problems

If your NFS client is having trouble performing a mount, try the following to verify that your VM system is running correctly:

1. Verify that the network connections are good. Try to "ping" the VM system, for example: ping gdlvmk1.endicott.ibm.com
2. Verify that the PORTMAPPER server is up and running by issuing the "RPCINFO" command. The RPCINFO command can be issued from VM or from another platform that supports it. From VM the command is issued as follows:

   ```
   RPCINFO -p server_name
   ```

   For example:

   ```
   RPCINFO -p gdlvmk1.endicott.ibm.com
   ```

   If the portmapper is up, a list of programs, versions, protocols, and port numbers is printed similar to the following:

   ```
   Ready; T=0.04/0.08 16:36:32
   rpcinfo -p k32lsf01
      program vers proto   port
       100000    2   udp    111  portmapper
       100000    2   tcp    111  portmapper
       100005    1   udp   2049  mountd
       100005    3   udp   2049  mountd
       100005    1   tcp   2049  mountd
       100005    3   tcp   2049  mountd
       100003    2   udp   2049  nfs
       100003    3   udp   2049  nfs
       100003    2   tcp   2049  nfs
       100003    3   tcp   2049  nfs
       150001    1   udp   2049  pcnfsd
       150001    2   udp   2049  pcnfsd
   Ready; T=0.04/0.08 16:38:21
   ```

   If a similar response is not returned, contact your system programmer or TCP/IP administrator to start the PORTMAPPER server on your VM system.
3. If your mount uses the NFS Version 3 or TCP transport protocol, verify that the output from the RPCINFO -p command lists **3** in the vers column and **tcp** in the proto column for both the **mountd** and **nfs** programs.
4. Verify that the **mountd**, **pcnfsd**, **portmapper**, and **nfs** services are running on your VM system by entering the following commands from VM:

```
rpcinfo -u server_name mount
rpcinfo -u server_name pcnfsd
rpcinfo -u server_name portmapper
rpcinfo -u server_name nfs
```

If the services are running on the VM system, the following responses are returned:

```
Ready; T=0.02/0.06 16:45:36
* See if mount is running
rpcinfo -u k32lsf01 mount
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
Ready; T=0.04/0.08 16:45:49

* See if portmapper is running
rpcinfo -u k32lsf01 portmapper
program 100000 version 2 ready and waiting
Ready; T=0.04/0.08 16:46:40

* See if nfs is running
rpcinfo -u k32lsf01 nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
Ready; T=0.04/0.08 16:47:27

* See if pcnfsd is running
rpcinfo -u k32lsf01 pcnfsd
program 150001 version 1 ready and waiting
program 150001 version 2 ready and waiting
Ready; T=0.04/0.08 16:47:53
```

If all of the above verifications look successful, the following can also be verified:

1. Contact your VM NFS server administrator to verify that the EXPORT and EXPORTONLY configuration statements in the VMNFS CONFIG file are set up correctly to allow your NFS client host system to perform the mount.

Errors Using Mounted Systems

Table 33 describes the status values that can be generated by the VM NFS server. Most of these status values are defined in RFC 1094 and RFC 1813. However, there are VM specific status values listed here also and there are status values listed that are defined in the RFCs but never returned.

The actual information that is visible to a program in the client system depends on the specific local file system routine invoked and the design of the NFS client implementation. Some NFS clients may make these error codes visible to the user, some do not.

Table 33. NFS System Return Codes

Code	Description
NFS3ERR_PERM = *1*	The caller does not have valid ownership, and the requested operation is not performed.
NFS3ERR_NOENT = *2*	The file or directory specified does not exist.
NFS3ERR_IO = *5*	I/O error occurred while an operation was in progress.
NFS3ERR_NXIO = *6*	Invalid device or address specified.
NFSERR_ENOMEM = *12*	The operation caused the server's file system to run out of memory. This status value is specific to the VM NFS server.

Table 33. NFS System Return Codes (continued)

Code	Description
NFS3ERR_ACCES = *13*	The caller does not have the correct authorization to perform the desired operation.
NFS3ERR_EXIST = *17*	The specified file already exists.
NFS3ERR_XDEV = *18*	An attempt was made to do a cross-device hard link. This status value is not returned by the VM NFS server.
NFS3ERR_NODEV = *19*	Invalid device specified.
NFS3ERR_NOTDIR = *20*	A non-directory was specified in a directory operation.
NFS3ERR_ISDIR = *21*	A directory was specified in a non-directory operation.
NFS3ERR_INVAL = *22*	An invalid argument or unsupported argument for an operation was received. An example is attempting a READLINK on an object other than a symbolic link.
NFS3ERR_FBIG = *27*	File too large. The operation caused a file to grow beyond the server's file system limit.
NFS3ERR_NOSPC = *28*	The operation caused the server's file system to run out of space.
NFS3ERR_ROFS = *30*	Writing attempted on a read-only file system.
NFS3ERR_MLINK = *31*	There are too many hard links.
NFS3ERR_NAMETOOLONG = *63*	The file name in an operation was too long, or invalid in some other way under the name translation option specified in the MOUNT command.
NFS3ERR_NOTEMPTY = *66*	Directory not empty. The caller attempted to remove a directory that was not empty.
NFS3ERR_DQUOT = *69*	Disk quota exceeded. The client's disk quota has been exceeded.
NFS3ERR_STALE = *70*	Invalid file handle. The file referred to by the file handle no longer exists, or access has been revoked. This access revoked condition also occurs under the following sequence of events. 1. A client mounts a CMS disk. 2. The minidisk link password granting access to that disk is changed. 3. The VM NFS server is restarted. 4. The client requests an operation on this disk. After it is restarted, the VM NFS server relinks a CMS disk when it first receives a client request referring to that disk. The server restart is not visible to the client. If the minidisk link password has been changed, this link fails and "stale filehandle" status is returned to the client. Similarly, if SFS authorization of BFS permissions change for a mounted directory, and attempts to access the directory fail following a server restart, "stale filehandle" status is returned.
NFS3ERR_REMOTE = *71*	Too many levels of remote in the path specified. The file handle given in the arguments referred to a file on a non-local file system on the server.
NFSERR_WFLUSH = *99*	Never returned by the VM NFS server.
NFS3ERR_BADHANDLE = *10001*	Never returned by the VM NFS server.

Table 33. NFS System Return Codes (continued)

Code	Description
NFS3ERR_NOT_SYNC = *10002*	Update synchronization mismatch was detected during a SETATTR operation.
NFS3ERR_BAD_COOKIE = *10003*	The cookie specified on READDIR or READDIRPLUS is incorrect.
NFS3ERR_NOTSUPP = *10004*	The operation specified is not supported.
NFS3ERR_TOOSMALL = *10005*	A buffer or request had an incorrect length.
NFS3ERR_SERVERFAULT = *10006*	Never returned by the VM NFS server.
NFS3ERR_BADTYPE = *10007*	An attempt was made to create an object of a type not supported by the server. For example, creating a socket using MKNOD is not supported for BFS.
NFS3ERR_JUKEBOX = *10008*	Never returned by the VM NFS server.

Notes for BFS Files and Directories

All functions defined by the NFS protocol are supported by BFS file systems.

1. By default, NFSERR_IO is returned on requests that reference file data for BFS files that have been placed in migrated status.
2. The size attribute is ignored on Create Directory.
3. All attributes are ignored on Create Link to File and Create Symbolic Link. Attributes for the new link match those of the linked-to file.
4. NFSERR_IO or NFS3ERR_INVAL is returned on all requests that attempt to use BFS external links of type CMSEXEC or CMSDATA. This restriction is in place because these types use ddnames and file modes, which will not be defined properly for the VM NFS server.
5. NFSERR_IO or NFS3ERR_INVAL is returned on all requests that attempt to read, write, or perform the set attributes set size function on a BFS FIFO.
6. All changes are flushed and committed for a Version 3 Commit request, even when offset and count contain non-zero values.
7. Both the VM user's primary GID (from the POSIXINFO statement) and the user's supplemantary GID list (from the POSIXGLIST statement) are used for requests.
8. The following restrictions apply to the MKNOD procedure:
   - creation of the block special device file and socket are not supported
   - only 16-bit major and minor device numbers are supported.
9. The file mode creation mask that is in effect for the VM NFS server machine will apply to the creation of BFS objects through the VM NFS server. In other words, the NFS client may request the permission values to be associated with the file or directory (based on a local client mask), but these may be overridden by the file mode creation mask on the VM NFS server.

   The file mode creation mask in effect can be displayed via OPENVM QUERY MASK. Your TCP/IP administrator can change the default by specifying the OPENVM SET MASK command when the VM NFS server starts. For more information on the file mode creation mask, see "Handling Security for Your Files" in the *z/VM: OpenExtensions User's Guide*, SC24-6206.
10. If the *requester* user ID specified on the mount (or with mountpw or pcnfsd) has file pool administration authority for the target file pool, that

administration authority is not respected by VMNFS. In other words, the *requester* user ID must have explicit permission to the object through the UID and GID associated with the user ID.

11. Refer to "VM NFS Server Link Support" on page 326 in this chapter for more notes on BFS symbolic links and external links.
12. If you are able to perform all expected functions for an NFS-mounted Byte File System directory, but ACCESS DENIED or NOT OWNER is displayed when you attempt to create a file, then the UID and GIDs defined for your client may be different from the UID and GIDs defined for the VM user ID used on the MOUNT. For a VM user ID, UID and GIDs are defined by the POSIXINFO, POSIXGROUP, and POSIXGLIST statements in the CP directory entry. Default values of -1 are used if these statements are not used.

    The file creation problem is typically seen on systems such as AIX and UNIX. The file is created successfully, but the NFS client then attempts to change the owning UID to match the UID of the VM user ID. This part of the operation fails because it requires super-user capabilities.

    Other types of clients tolerate differences in UID/GID definitions more easily. These other NFS clients often use PC-NFS to ask the server: `By what UID am I known here?` The NFS client then uses that information by recognizing what the owning UID of a new file should be.

    How can you correct this problem on a system such as AIX?

    a. The best way to avoid this problem is to have global user ID definitions. The VM user ID is assigned the same UID and GIDs used by the NFS Client system.
    b. If global user ID definitions are not possible, another choice is to create a new user ID on the client system with a UID that matches the VM user ID. Use 'su' to switch to that user ID on the client before creating the file. Also ensure that the GID assigned to the new name is in the list of GIDs assigned to the VM user ID. This may be easier than forcing consistency across all systems, but it is practical only if there is no overlap in the UID definitions for the two systems.
    c. A third choice is to MOUNT using a VM user ID that has super-user authority. This will allow the create request to succeed. In this case, clients that use the mount point have full power to anything under it (because they are considered super-users by the VM NFS server). Access to the mount point must be carefully controlled from the client side.

Notes for SFS Files and Directories

1. Most functions defined by the NFS protocol are supported by SFS file systems. Those not supported include:

    - symbolic links

    Those partially supported include:

    - `link`

      Limited support is available for the link NFS request. The intent of this support is to accommodate clients that use link as a temporary step during a file rename or similar operation. A hard link may be created only in the same directory as the source file. Link information is maintained in volatile storage by the VM NFS server, and it is not used when the reply to a read-directory or lookup-file request is constructed.
    - set-attributes
      – Both the time of last modification (`mtime`) and the time of last access (`atime`) can be changed. However, SFS maintains only the date, (not time)

of last access for SFS files, so when an `atime` is retrieved, it will show a timestamp that is midnight of the given date.
  - Requests to change file permissions are ignored and *success* status is returned to the client.
  - Requests to change `mode`, `uid`, or `gid` are ignored and *success* status is returned to the client.
  - Attempting to increase the size of a variable file generates an error response.
- Only the file space owner may create (`mkdir`) or remove (`rmdir`) directories, unless an ESM is used and allows the operation for the requester user ID.
- The file size attribute can be used on Create File to set the size for fixed record format files. The file size attribute is ignored and set to zero for variable record format files.

2. Access to SFS file and directories is based on the authorities granted to the *requester* user ID specified on the MOUNT command, and not UID's. On multiple user systems such as UNIX, all users have the same access to objects in the mounted SFS directory if they have permission to use the mount point. Thus the administrator (super-user) who performs the mount must make sure that appropriate controls are in place for the mount point.
3. Each update operation is committed before responding to the client, including Version 3 Write requests. A Version 3 commit request returns a **success** status to the client.
4. CMS users can create SFS files that do not map to NFS file types. The `ftype` value returned for aliases, erased aliases, revoked aliases, and external objects is NFNON (non-file).
5. When names=Trans is specified on the Mount command, NFS provides translation of SFS directory names that are greater than 16 characters or contain invalid characters. There is one restriction: `mkdir`() or `rename`() are rejected if the name contains a dot.

   The `mkdir`() operation creates FILECONTROL directories and uses the `mtime` (modification time), if provided by the client.

   The only date used for SFS directories is mdate.
6. A **ro** (read-only) mount request is similar to the RORESPECT ON setting in a CMS client. If an SFS directory is mounted **ro**, you may not write to files in that subdirectory structure.
7. By default, NFSERR_IO is returned on requests that reference file data for SFS files that have been placed in migrated status.
8. Files named ##NFS## #VHIST# and ##NFS## #NAMES# may reside in an SFS top directory when files are written using NFS. Do not modify or erase these files. They contain control information needed by NFS. Do not copy the ##NFS## #NAMES# file. If you are using name translation and want to move a file from one file system (such as an SFS directory) to a separate file system (such as a minidisk), NFS mount both file systems and copy the file. This allows the VMNFS file server to proper format.
9. Not all file attributes defined by the NFS protocol apply to SFS file systems. In the case of `uid` and `gid`, the attribute values are returned as follows:
   - The `uid` value returned for an SFS Dircontrol directory will be 1. All other SFS objects will return a `uid` value of 0.
   - The `gid` value for an SFS file with fixed record format will be returned as 1. The `gid` value for an SFS file with variable record format will be returned as 2. All other SFS objects will return a `gid` value of 0.

- The record length of an SFS file (LRECL) will be returned in the `rdev` attribute.

10. Creation of special files using MKNOD is not supported. Special files are block special device file, character special device file, socket, and named pipe.
11. Creation of a regular file with the mode set to EXCLUSIVE is not supported.
12. Records in CMS variable files are limited to a length of 65535. If a write request attempts to create a record larger than this, a message is written to the VM NFS server console and an I/O error is returned to the client.
13. The NFS READDIR procedure can appear to behave inconsistently with NFS mounted SFS directories, due to native SFS authorization rules. SFS will allow non-directory file system objects to be listed in NFS READDIR output, provided that the NFS client has at least SFS read authorization to the directory containing the objects, but SFS will not allow subdirectories in that directory to be listed in NFS READDIR output, unless the NFS client was also explicitly given at least SFS read authorization to each subdirectory.

    To ensure that clients will be able to see SFS subdirectories after an NFS READDIR procedure, SFS read authorization should be granted for every directory in a file system hierarchy. Refer to the *z/VM: CMS User's Guide*, SC24-6173, for more information on SFS authorizations.
14. When writing to an existing SFS fixed format file for a mount point created using the **lines=nl** option, the data must be formatted so that the **NLvalue** follows every *nn* bytes of data, where *nn* is the logical record length for the file. If no **NLvalue** is found where one is expected, an I/O error is returned to the client. One workaround for this restriction is to use the "save as" function in your editor to save the file under a new name. By default, when using the **lines=nl** option, a new file is created in the variable file format.
15. If two **NLvalues** are provided in a row when writing to an SFS variable format file, the second **NLvalue** will appear to be at the start of the following record when viewing the file from CMS. Zero-length records are not allowed in the CMS record file system.

Notes for Minidisk Files

1. Some functions that are defined by the NFS protocol are not supported by CMS minidisk file systems. Others are partially supported.

    Those not supported include:
    - make-directory
    - symbolic link
    - remove-directory

    Those partially supported include:
    - set-attributes
      - Only the time of last data modification (`mtime`) can be changed.
      - Requests to change file permissions are ignored and *success* status is returned to the client.
      - The size of a file can be changed to zero or to its current size. The size of a fixed record format file can be increased. Other size change requests generate an error response.
      - Requests to change `mode`, `uid`, or `gid`, are ignored and *success* status is returned to the client.
    - link
      - Limited support is available for the link NFS request. The intent of this support is to accommodate clients that use link as a temporary step during a file rename or similar operation. Link information is maintained

in volatile storage by the VM NFS server, and it is not used when the reply to a read-directory or lookup-file request is constructed.

- Access to minidisks is based on the ability of the VM NFS server to link the disk. On multiple user systems such as UNIX**, all users have the same access to files in the mounted minidisk if they have permission to use the mount point. Thus the administrator (super-user) who performs the mount must make sure that appropriate controls are in place for the mount point.
- A minidisk file size can be changed to zero from an NFS client. To NFS clients it will appear to have a file size of zero; to CMS users the file will appear to have one record, of length one.
- Not all file attributes defined by the NFS protocol apply to minidisk files. In the case of `uid` and `gid`, the attribute values returned are as follows:
  - The `uid` value returned for a minidisk file with either fixed or variable record format will be returned as 0.
  - The `gid` value for a minidisk file with fixed record format will be returned as 1. The `gid` value for a minidisk file with variable record format will be returned as 2.
  - The record length of a minidisk file (LRECL) will be returned in the `rdev` attribute.
- The VM NFS server updates CMS minidisk disk blocks in place, if possible. The advantages of this implementation include:
  - Updates to existing data in a file are immediately visible to a CMS user who accessed the disk before the updates were made.
  - The VM NFS server does not need to keep elaborate data structures in memory, record the current state of a file, or manipulate the CMS allocation and directory information every time a write request is executed.

  The disadvantage of this type of implementation is that a time window exists, during which a failure of the VM NFS server could cause a disk block on a CMS disk to be recorded as allocated, when it does not belong to any file.

  This time window is of short duration, because, after the block is marked as allocated, the directory or pointer block necessary to record ownership of the block is immediately updated. For this reason, troubles with lost blocks are infrequent.

  If a file is open when an update is made, you may not see the update, because CMS buffers the disk block containing the update. CMS only buffers one disk data block for each file, so updates in another data block can be seen, even if the file is open. Closing the file makes the updates visible the next time the file is read.

  Updated file data is immediately visible to other NFS clients, subject to the limitations imposed by the buffering performed in those client machines.

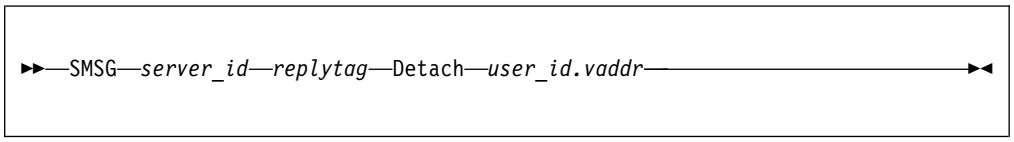
  Write operations performed by the VM NFS server are similar to the operations that CMS performs to files that have the file mode number 6. The VM NFS server does not distinguish the actual file mode number associated with a file on a CMS disk. New files are always created with the file mode number 1. Files of any file mode number (including zero) can be read by any client that performs a successful mount.
- The file size attribute can be used on Create File to set the size for fixed record format files. The file size attribute is ignored and set to zero for variable record format files.

2. All changes are flushed and committed for a Version 3 Write request. A Version 3 Commit request returns a **success** status to the client.
3. Creation of special files using MKNOD is not supported. Special files are block special device file, character special device file, socket, and named pipe.
4. Creation of a regular file with the mode set to EXCLUSIVE is not supported.
5. Records in CMS variable files are limited to a length of 65535. If a write request attempts to create a record larger than this, a message is written to the VM NFS server console and an I/O error is returned to the client.
6. When writing to an existing minidisk fixed format file for a mount point created using the **lines=nl** option, the data must be formatted so that the **NLvalue** follows every *nn* bytes of data, where *nn* is the logical record length for the file. If no **NLvalue** is found where one is expected, an I/O error is returned to the client. One workaround for this restriction is to use the "save as" function in your editor to save the file under a new name. By default, when using the **lines=nl** option, a new file is created in the variable file format.
7. If two **NLvalues** are provided in a row when writing to a minidisk variable format file, the second **NLvalue** will appear to be at the start of the following record when viewing the file from CMS. Zero-length records are not allowed in the CMS record file system.
8. Files named ##NFS## #VHIST and ##NFS## #NAMES# may reside on a minidisk when files are written using NFS. Do not modify or erase these files. They contain control information needed by NFS. Do not copy the ##NFS## #NAMES# file. If you are using name translation and want to move a file from one file system (such as a minidisk) to a separate file system (such as an SFS directory), NFS mount both file systems and copy the file. This allows the VMNFS file server to create a mapping for the client file name in the proper format.

SMSG Interface to VMNFS

The following are the SMSG commands supported by the NFS server.

SMSG DETACH Command



Purpose

Use the SMSG DETACH command to direct the NFS server to detach a minidisk that it has linked.

Note: Some SMSG commands may be restricted by the TCP/IP administrator for use by only authorized VM user IDs.

SMSG command operands are parsed by the NFS server as blank-delimited tokens, and case is not significant. Any tokens present after those recognized by the NFS server are considered to be comments and are ignored.

Operands

server_id
The user ID of the NFS server virtual machine, usually VMNFS.

replytag
A character string that associates the supplied SMSG command *Option* with a server response; the *replytag* prefaces each message issued by the NFS server in response to the given command. Any characters (other than blank and null characters) can be used to form a *replytag*, and case is not significant.

The *replytag* also indicates how the NFS server is to deliver its response to a particular SMSG command. The last character of *replytag* has special meaning and is interpreted by the server as follows:

The following describes how the last character of *replytag* is interpreted.

s respond using the CP SMSG command.

m respond using the CP MSG command.

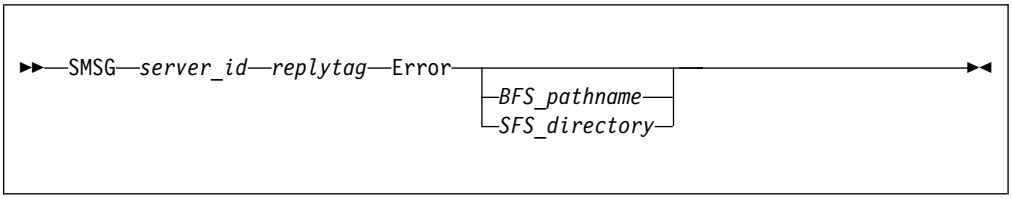
n send no response.

If the *replytag* does not end with one of these characters, the NFS server chooses a response mode.

user_id
Identifies the VM user ID that owns the minidisk.

vaddr
Is the minidisk virtual address.

SMSG ERROR Command



Purpose

Use the SMSG ERROR command to obtain error information about client requests that pertain to a specific SFS or BFS directory that has been mounted.

Note: Some SMSG commands may be restricted by the TCP/IP administrator for use by only authorized VM user IDs.

SMSG command operands are parsed by the NFS server as blank-delimited tokens, and case is not significant. Any tokens present after those recognized by the NFS server are considered to be comments and are ignored.

Operands

server_id
The user ID of the NFS server virtual machine, usually VMNFS.

replytag
A character string that associates the supplied SMSG command *Option* with a server response; the *replytag* prefaces each message issued by the NFS server in

response to the given command. Any characters (other than blank and null characters) can be used to form a *replytag*, and case is not significant.

The *replytag* also indicates how the NFS server is to deliver its response to a particular SMSG command. The last character of *replytag* has special meaning and is interpreted by the server as follows:

The following describes how the last character of *replytag* is interpreted.

s respond using the CP SMSG command.

m respond using the CP MSG command.

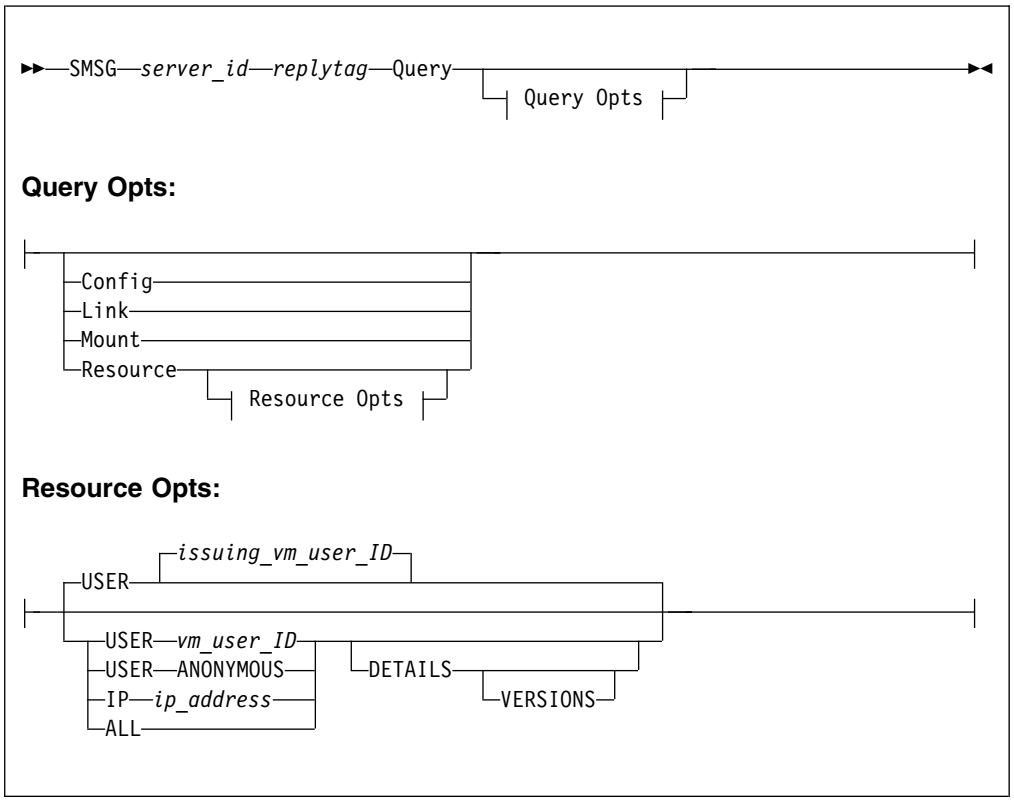
n send no response.

If the *replytag* does not end with one of these characters, the NFS server chooses a response mode.

BFS_pathname
The mounted BFS directory for which host error information is requested. Information will only be returned if the VM user ID that originates the error request has permission to the directory in question.

SFS_pathname
The mounted SFS directory for which host error information is requested. Information will only be returned if the VM user ID that originates the error request has authorization for the directory in question.

SMSG QUERY Command



Purpose

Use the SMSG Query command to obtain a summary of NFS server activity.

Note: Some SMSG commands may be restricted by the TCP/IP administrator for use by only authorized VM user IDs.

SMSG command operands are parsed by the NFS server as blank-delimited tokens, and case is not significant. Any tokens present after those recognized by the NFS server are considered to be comments and are ignored.

Operands

server_id
The user ID of the NFS server virtual machine, usually VMNFS.

replytag
A character string that associates the supplied SMSG command *Option* with a server response; the *replytag* prefaces each message issued by the NFS server in response to the given command. Any characters (other than blank and null characters) can be used to form a *replytag*, and case is not significant.

The *replytag* also indicates how the NFS server is to deliver its response to a particular SMSG command. The last character of *replytag* has special meaning and is interpreted by the server as follows:

The following describes how the last character of *replytag* is interpreted.

s respond using the CP SMSG command.

m respond using the CP MSG command.

n send no response.

If the *replytag* does not end with one of these characters, the NFS server chooses a response mode.

Query
Requests summary information about NFS server activity. The QUERY command can be further qualified with the CONFIG, LINK, MOUNT or RESOURCE operands to obtain specific information.

Query CONFIG
Requests information about how the NFS server is configured. The server returns the following information in its response:

- The TCP/IP function level of the VMNFS MODULE in use.
- The status of various configuration parameters, such as whether mounts are restricted to only exported file systems, whether anonymous mounts are allowed, whether PCNFSD is supported, and the time after which PCNFSD information will be discarded.
- The maximum buffer size to be used to satisfy NFS version 3 READ requests.
- The maximum buffer size to be used to satisfy NFS version 3 WRITE requests.
- The maximum number of NFS clients that can concurrently use the TCP transport protocol.
- The number of NFS clients that are concurrently using the TCP transport protocol.

Information returned by the SMSG QUERY CONFIG command can be used to determine if the maximum allowable number of clients which use the TCP transport protocol (controlled by the MAXTCPUSERS server configuration statement) is acceptable.

Query Link
Requests information about minidisks that are linked by the NFS server. Support for the SMSG QUERY LINK command is provided to maintain compatability with prior levels of TCP/IP for VM/ESA; it is recommended that the SMSG QUERY RESOURCE command be used in place of the SMSG QUERY LINK command.

Query Mount
Requests information about NFS mounts that are active. Support for the SMSG QUERY MOUNT command is provided to maintain compatability with prior levels of TCP/IP for VM/ESA; it is recommended that the SMSG QUERY RESOURCE command be used in place of the SMSG QUERY MOUNT command.

Query Resource
Requests information about specific resources that are actively in use by the NFS server. SFS and BFS directories are considered to be active if they have been used or referenced within the 15 minute period that percedes receipt of an SMSG QUERY RESOURCE command. A minidisk is considered to be active so long as the NFS server has established a link to that minidisk.

If additional operands are not used to qualify a RESOURCE query, information that corresponds to the user ID which originated the SMSG command is returned, as if the USER *vm_user_id* had been specified.

The information returned in response to an SMSG QUERY RESOURCE command includes:

- whether a mount is read-write or read-only.
- a one-character mount *indicator*. This indicator is typically **m**, meaning the returned resource information corresponds to an explicit client mount. An asterisk (*) indicates the NFS server was restarted and that an implicit mount was performed for a directory that is lower in the hierarchy of the explicitly-mounted directory.
- the IP address from which a mount was requested.
- the VM user ID under which a mount was requested, or "anonymous" if an anonymous mount was performed.
- the name of the mounted BFS directory, SFS directory, or minidisk.

The RESOURCE operand can be further qualified with the USER, IP or ALL operands to obtain more specific summary information, as follows:

USER *vm_user_id*
Limits resource information for directories and minidisks mounted under authority of the specified VM user ID, *vm_user_id*.

USER ANONYMOUS
Limits resource information for directories and minidisks that have been mounted anonymously.

IP *ip_address*
Limits resource information for directories and minidisks that have been mounted by the host with the IP address, *ip_address*.

ALL Provides resource information for all mounted directories and minidisks. Note that BFS and SFS directory mount information is returned only if the user ID that originates the request has permission (authorization) for those directories. Table 34 on page 323 shows the information that is displayed when the ALL operand is used.

Table 34. Resource Information

Mount Information	Meaning
*	The NFS server was restarted and an implicit mount was done for a directory lower in the explicitly-mounted directory's hierarchy.
m	Information displayed is for an explicit client.
ro, **rw**	Indicates whether the mount is read-write or read-only.
name	The name of the mounted BFS directory, SFS directory, or minidisk.
IP address	The IP address from which the mount was requested.
user ID	The VM user ID under which the mount was requested. This will contain “anonymous” if the mount was made anonymously.

The operands that follow can be used with the USER, IP or ALL operands to obtain additional information, as follows:

DETAILS
Provides counter values for various NFS requests.

VERSIONS
Provides separate request counter values for NFS version 2 requests and NFS version 3 requests.

Examples

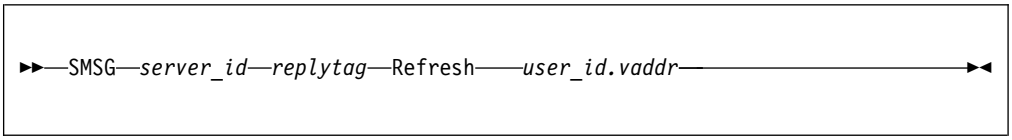
- The following example asks for a display of NFS server activity.

  ```
  smsg vmnfs m q
  ```

  The reply from this QUERY command is sent by the CP MSG command. The message text (time and origin information added by CP is omitted) that might result from this example is:

  ```
  M VM NFS server start time 18Jan1999 08:50:48.
  M 238 RPC (0 duplicate XID), 11 SMSG, 5 *BLOCKIO
  M 0 null, 6 getattr, 7 setattr, 98 lookup, 12 read, 4 write
  M 4 create, 3 remove, 2 rename, 0 link, 70 readdir, 14 statfs
  M 13 mkdir, 1 rmdir, 0 symlink, 0 readlink, 0 access, 0 mknod
  M 0 readdir+, 0 fsstat, 0 fsinfo, 0 pathconf, 0 commit
  M 2 mount, 0 mountpw, 1 mountnull, 0 unmount, 0 unmount-all
  M 1 pcnfsd auth, 0 unsupported
  M End of reply.
  ```

SMSG REFRESH user_id.vaddr Command



Purpose

Use the SMSG REFRESH user_id.vaddr command to refresh minidisk information after changes have been made to a minidisk that the server has linked in read-only mode; this causes buffered disk block data to be discarded so that subsequent client requests make use of *current* disk data.

Note: Some SMSG commands may be restricted by the TCP/IP administrator for use by only authorized VM user IDs.

SMSG command operands are parsed by the NFS server as blank-delimited tokens, and case is not significant. Any tokens present after those recognized by the NFS server are considered to be comments and are ignored.

Operands

server_id
The user ID of the NFS server virtual machine, usually VMNFS.

replytag
A character string that associates the supplied SMSG command *Option* with a server response; the *replytag* prefaces each message issued by the NFS server in response to the given command. Any characters (other than blank and null characters) can be used to form a *replytag*, and case is not significant.

The *replytag* also indicates how the NFS server is to deliver its response to a particular SMSG command. The last character of *replytag* has special meaning and is interpreted by the server as follows:

The following describes how the last character of *replytag* is interpreted.

s respond using the CP SMSG command.

m respond using the CP MSG command.

n send no response.

If the *replytag* does not end with one of these characters, the NFS server chooses a response mode.

user_id
Identifies the VM user ID that owns the minidisk.

vaddr
Is the minidisk virtual address.

Examples

- The following example instructs the NFS server to discard any disk blocks buffered from the TCPMAINT 592 minidisk.

  ```
  smsg vmnfs example1m refresh tcpmaint.592
  ```

  The reply from the SMSG command is sent by the CP MSG command. The message text returned for this command is:

  ```
  EXAMPLE1M REFRESH TCPMAINT.592 completed.
  ```

  If a busy reply is sent, the message text is:

  ```
  EXAMPLE1M REFRESH TCPMAINT.592 busy.
  ```

  The busy reply is generated if the VM NFS server cannot find a moment when a client request is not using the referenced minidisk. The server tries for 10 seconds before generating a busy reply. If a busy reply is received, you can try the command again.

Name Translation File

The VM NFS server creates the translation file ##NFS## #NAMES# on a:

- CMS minidisk, or
- SFS top directory

that has been mounted read-write with default name processing when the first client request to create an invalid file name for CMS is received. The translation file contains both the original client file names and the corresponding CMS file names that are generated.

The names translation files are hidden from NFS clients. That is, lookup or readdir requests do not return an entry for these files. Please note that these files are only visible to CMS users of the minidisk or directory.

Attention
You can destroy a name translation file by writing incorrect data to the file. This damages the internal structure of the file. You can also destroy a name translation file by erasing or renaming it using CMS.

Special File Names for VM NFS

The file names (.) and (..) are handled as special cases by the VM NFS file server. No name translation is performed. For minidisk, the file lookup procedure in the server treats each of these file names as a request for the file handle of the directory (the CMS minidisk itself). For SFS and BFS, a file name of (..) is treated as a request for a lookup of the parent directory of the current object. A file name of (.) is treated as a request for a lookup of the current object. This convention supports interpretations that are common in UNIX NFS clients.

Special File Name Considerations

Note for ESM-protected file pools: an NFSERR_ACCES error code may indicate that you do not have the correct authorization to perform the desired operation on the target file, *or* on the name translation file that resides in an SFS top directory.

Note to UNIX users: CMS file rename semantics differ from UNIX. CMS uses file names to denote files, unlike the inode organization that UNIX uses for its files. Changing the name of a file for which some client has a file handle makes that file handle invalid, because there is no longer a file with the name denoted by the file handle. If a new file is created with the old name, or an existing file is renamed to the old name, the old file handle denotes the new file.

UNIX systems avoid a similar problem when reusing inodes, because they maintain an inode generation number which, in combination with the inode number, uniquely identifies a file. A new file with the name that another file used to have is different from the old file, because the inode number and generation number of the new file is different from the old one.

NFS Client Problems

Some client implementations have a particular problem with deleting files from CMS minidisks. The problem results from a discrepancy between assumptions made by the client about how the VM NFS server manages a directory, and the actual mechanism used by CMS to manage the directory on a minidisk. The problem occurs when the PC-DOS client undertakes an operation such as `delete *.c`. This operation starts with the client reading a buffer of file names (read-directory operation) from the server, and receiving a `cookie` (file name) that references a particular position in the directory. The client then examines the names, finds one or more that match the specified pattern, and emits erase-file calls for the matching names. Because CMS maintains a compact directory, the hole

created by deleting a file is filled by moving the data from the last file in the directory into the hole and shortening the length of the directory itself.

When the client finishes erasing all appropriately named files in this first group of file names, it calls the server (using the previously obtained `cookie`) to read another group of file names. However, a number of file names (equal to the number of erased files) can no longer be seen by the client, because the directory is not the same as when the first group of file names was read. In this example, if one of the holes was filled with data for a file that matches the "`*.c`" pattern, one or more files that should have been erased are retained.

A similar problem between the client and server, again due to a cookie-related discrepancy, can occur with BFS and SFS managed directories, with some NFS client implementations.

You can avoid these problems by making the client application repeat the delete request, if it deletes any files using a wildcard pattern, until it receives a return value indicating that no file names were matched by the pattern. For more specific instructions for particular clients please refer to the VM TCP/IP homepage at IBM: TCP/IP for z/VM and VM/ESA (www.ibm.com/vm/related/tcpip/).

Deleting CMS Record-Length Fields

In the past, the VCMS C sample program was provided to read variable format (V-format) files and delete CMS record-length fields. This sample is no longer provided. Instead the **lines** option can now be specified on MOUNT requests to prevent CMS record-length fields from being inserted in the NFS data stream (if CMS record-length fields are not needed).

Using NFS with RACF

The Resource Access Control Facility (RACF) allows NFS servers to act as *surrogates* for other user IDs. This means that the server can access those disks available to a given user ID.

The command that allows NFS servers to act as surrogates is provided in a program called NFSPERM EXEC. To use it, enter the command:

```
NFSPERM ADD
```

If you get an error, contact your system administrator.

You may delete the NFS server's surrogate authority by issuing the command:

```
NFSPERM DELETE
```

NFSPERM is explained in the "Using TCP/IP with External Security Manager" section of the *z/VM: TCP/IP Planning and Customization* publication.

VM NFS Server Link Support

Two kinds of links are supported in the NFS protocol, hard links and symbolic links. Hard links are supported by the NFS LINK procedure, and symbolic links are supported by the NFS SYMLINK and READLINK procedures. VM does not natively support hard links and symbolic links for minidisk file systems and SFS, but there is native support for hard links and symbolic links in BFS.

SFS and Minidisk Links

The VM NFS server provides no additional support for symbolic links on minidisk or in SFS, but it does provide limited support for minidisk and SFS hard links. This support is minimal, and is intended to facilitate some NFS clients' intermediate use of hard links during NFS operations such as RENAME.

BFS Links

The VM NFS server fully supports hard links and symbolic links in BFS. A symbolic link may appear in NFS READDIR output, and may be a component of a BFS pathname on an NFS MOUNT command or other NFS client commands.

The VM NFS server provides limited support for BFS external links, even though these fall outside the scope of the NFS protocol. MOUNT external links may appear in NFS READDIR output, may be targets of NFS READLINK procedures, and may be components of BFS pathnames on an NFS MOUNT command or other NFS client commands. NFS READLINK does not support CMSDATA, CMSEXEC, or CODE external links, but they may appear in NFS READDIR output. Creation of external links, and I/O operations on external links, are not supported by the VM NFS server.

Because symbolic links and MOUNT external links may be BFS pathname components, seemingly inconsistent behavior related to pathname parsing can occur during some NFS operations. NFS clients interpret the contents of symbolic links and MOUNT external links when they are encountered as pathname components, and different NFS clients may interpret them differently; this can affect the outcome of NFS READDIR operations and other NFS operations.

- The use of consecutive periods (**..**) in a BFS pathname can cause problems. UNIX-type implementations interpret this to mean the parent directory of the current directory, and it may be meaningless in non-UNIX-type implementations.
- The VM OpenExtensions fully-qualified BFS prefix "/../VMBFS:" can be a problem if it appears other than at the beginning of a pathname.
- A slash (/) at the beginning of a BFS pathname can imply that the subsequent path is relative to the root of the current file system, and the absence of a slash at the beginning can imply that the subsequent path is relative to the current directory.
- A slash (/) at the end of a pathname can imply that a link should be followed, and the absence of a slash at the end can imply that a link should not be followed.

The following recommendations should thus be observed:

- Consider the run-time perspective of the NFS client when constructing pathnames for symbolic links and MOUNT external links in BFS.
- Do not use consecutive periods (**..**) in a BFS pathname unless its meaning will be unambiguous at run time.
- Be conscious of whether links are being specified relative to the expected current directory at run time, or relative to the root of the file system.

Refer to the *z/VM: OpenExtensions User's Guide* and the *z/VM: OpenExtensions Commands Reference* for more information on BFS symbolic links, external links, and pathname syntax and resolution.

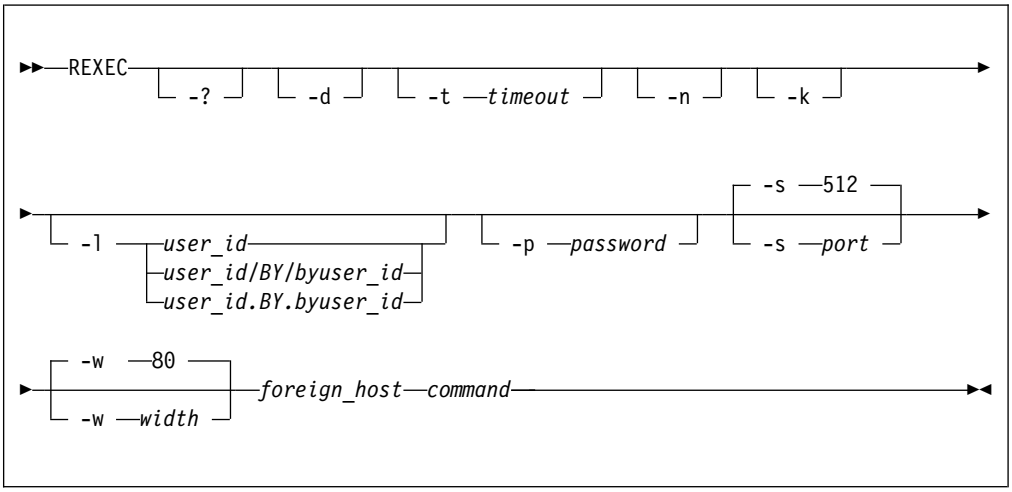
Chapter 10. Using the Remote Execution Protocol

The Remote Execution Protocol (REXEC) is a remote execution client that allows you to execute a command on a foreign host and receive the results on the local host. This chapter describes how to use the REXEC command, and its associated NETRC DATA file.

The REXEC client passes a user name, password, and command to an REXEC daemon on a foreign host, which provides automatic logon and user authentication, depending on the parameters set by the (client) user. If the authentication process succeeds, the command is issued, and any results are returned to the client. If the authentication fails, an error message is displayed on the local host.

An REXEC daemon must be running on the foreign host where your command is to be issued. For information about the TCP/IP REXEC daemon (REXECD), see *z/VM: TCP/IP Planning and Customization*. For information about REXEC daemons on other platforms, see the appropriate documentation associated with the foreign host being used.

REXEC Command



Purpose

Use the REXEC command to execute commands on a foreign host and receive the results on the local host.

Operands

REXEC operands are case-sensitive.

-? Displays the help message.

-d Activates debug tracing.

© Copyright IBM Corp. 1987, 2016

-t *timeout*
Sets idle time-out. This option specifies the time (in seconds) in which a connection closes if there is no activity. By default, the session will remain active until the remote system closes the connection.

-n Suppresses use of the NETRC DATA file and hence automatic logon to the foreign host.

-k Suppresses prompts for the logon user name and password. Use this option when a NETRC DATA file is used and command input is to be obtained through non-interactive means, such as from an exec.

-l *user_id*
Specifies a user ID that is valid on the foreign host. Depending on the type of host, *user_id* may be case sensitive.

-l *user_id/BY/byuser_id*
-l *user_id.BY.byuser_id*
Specifies an alternate logon name to be used by a foreign z/VM host during authorization. When the /BY/ or .BY. delimiter and the *byuser_id* parameter are specified with *user_id*, the *byuser_id* logon password is used for authorization checking instead of the *user_id* password.

Any user ID specified in the *byuser_id* parameter must be included in a LOGONBY statement in the *user_id* CP directory entry. When an External Security Manager (ESM) such as RACF is installed, its defined authorization criteria may override that defined by CP in the LOGONBY statement. For example, RACF may perform authorization checks for attempts to log on a shared user ID. For further details, refer to the documentation for the ESM in use.

-p *password*
Specifies the password associated with *user_id*. The password may be case sensitive. If the password contains blanks, it must be surrounded by single quotation marks and any single quotation marks within it must be doubled.

-s *port*
Specifies the TCP port number of the REXEC daemon on the foreign host. The default for *port* is 512.

-w *width*
Specifies the maximum width to use for lines written by REXEC. The minimum acceptable width is 1, while the maximum is 32767. The default for *width* is 80. Note that circumstances on the remote host continue to affect or restrict output when the **w** operand is used.

foreign_host
Specifies the name or internet address of the foreign host to which you are sending your command. You can specify the foreign host by its host name or internet address.

command
Specifies the command that is sent to the foreign host. The *command* can be composed of one or more words. After checking for any prefixed parameters (-l, -p, and -s) and obtaining the foreign host, the remaining REXEC argument string is assigned to *command*. Depending on the type of foreign host, the command may be case sensitive.

Usage Notes

1. If you omit the user name or password, REXEC prompts you to supply them, unless they are defined in a NETRC DATA file, or unless the **-k** option has been specified to suppress prompting.

Examples

- The following example shows the results obtained for an REXEC command issued against another VM system. In this example, the CP QUERY NAMES command has been executed by an rexec daemon agent (RXAGENT1) on host ODDJOB, as a result of a user specifying a user ID and password of "GUEST".

```
rexec -l guest -p guest -s 512 oddjob cp q names
VMNFS    - DSC, SMTP      -DSC,  PORTMAP   - DSC
LPSERVE  - DSC, FTPSERVE  -DSC,  REXECD    - DSC
SNMPQE   - DSC, SNMPD     -DSC,  RSCS      - DSC
TCPIP    - DSC, PVM       -DSC,  GCS       - DSC
OPERSYMP - DSC, DISKACNT  -DSC,  EREP      - DSC
TCPMAINT - DSC, RXAGENT1  -DSC,
VSM      - VTAM
VSM      - TCPIP
Ready;
                                                    RUNNING   GDLVM7
```

The NETRC DATA File

The NETRC DATA file provides an alternative for specifying the REXEC *user_id* and *password* parameters. If these parameters are defined within this file for a specific host, they can be omitted when you issue an REXEC command against that host.

For more information on the NETRC DATA File and how it may be used with other applications, see Appendix B, "Using the NETRC DATA File," on page 453.

The following sections should be reviewed before REXEC commands are issued against a VM host running TCP/IP, as they provide important information about the REXEC support provided in the VM environment.

Anonymous Remote Command Execution

TCP/IP provides support for "anonymous" REXEC command execution through the use of one or more REXEC "agent" machines. These machines can be used by specifying either of the following user ID and password combinations:

User ID
: **Password**

ANONYMOU
: ANONYMOU

GUEST
: GUEST

Note: Only the first eight characters of the above values are verified. For the case when ANONYMOU is used, longer strings that begin with "ANONYMOU" (such as "ANONYMOUS") may be accepted. Also, note that these values are not case-sensitive.

When an agent machine is used to execute an anonymously-supplied command, the command is passed on to, and executed within, an available agent machine. After the command completes, output is obtained by REXECD via IUCV

communications and is passed on to the REXEC client; the agent machine is then made available to process other anonymous REXEC commands. Note that since multiple agent machines may be in use on the remote VM host, there is no guarantee that the same agent will be used to process successive REXEC commands. Also, since a given agent machine handles multiple users' requests, the internal state of that machine may vary as different commands are issued.

Command Execution Using Your Own Virtual Machine

A CMS user's own virtual machine can also be used to execute REXEC-supplied commands. When this is done, command processing is performed in much the same way as that done with RXAGENT machines. The main difference is that your virtual machine is autologged following user and password authentication, and then logged off after the given command has completed.

It should be noted that if a client application sends consecutive REXEC commands in quick succession to the VM TCP/IP REXECD server machine, one (or more) of these commands may fail due to errors associated with logging on the intended virtual machine. For example, the message "Unable to autolog user *user ID*" may be received. Errors such as this may occur due to timing conflicts between the autolog processing and that of the supplied commands. This situation can be avoided by enlisting the use of an RXAGENT machine or by adding an appropriate delay between consecutive commands issued by the client.

In addition to the items listed in the Notes and Restrictions section, the following considerations apply to any CMS user's virtual machine used for REXEC command processing:

1. The user machine must not be active when REXEC requests are made against it. If the user's machine is logged on or disconnected, the REXEC request will be rejected.
2. Any machine used to process REXEC commands must access the TCP/IP client minidisk TCPMAINT 592, in its PROFILE EXEC. This is necessary so that the RXSNDIU EXEC is available, which is used to process data provided by REXECD after the machine has been autologged.
3. Your PROFILE EXEC should not attempt to process data in the console input buffer or program stack. Such processing, if performed, may prevent REXEC commands from being processed. Additionally, the PROFILE EXEC must not require user intervention.

Notes and Restrictions

1. Due to the manner in which command output is obtained from agent machines, secondary consoles must not be defined for them. If a secondary console is defined, REXECD will not be able to send command results back to the REXEC client.
2. When the REXECD server autologs a virtual machine, it passes several parameters. In doing so, the # (pound) symbol is always used as the LINEND character. If a different LINEND character is in effect for the agent machine(s), REXEC commands will not be processed successfully. In such cases, the agent machine's PROFILE EXEC will need to be modified to set the LINEND character to #, with the CP TERMINAL command. For example, when the foreign host is a VM system you would enter:

   ```
   CP TERMinal LINEND #
   ```

3. Because the REXEC protocol doesn't allow for user interaction, commands to be executed remotely must complete without the need for user intervention. When a VM system is the foreign host involved, commands such as FILELIST should not be executed via REXEC.
4. The combined length of all REXEC operands, including spaces, is limited to 255 bytes. Therefore, the number and length of the operands that precede the *command* operand (such as -d or -l *userid*) will further restrict the length of a command that can be sent to a foreign host.

   Commands submitted to remote VM hosts are further limited, because of REXECD processing constraints. Commands that are processed anonymously by an RXAGENT maachine are limited to 227 bytes; those processed by a CMS user's own virtual machine are limited to 189 bytes.

Using RSH commands with REXECD

TCP/IP provides limited support for **rsh**, **remsh**, and similar commands. When such commands are issued against a VM host, a command to be executed on the remote VM host must be supplied. There is no support for the processing of interactive commands.

When an rsh command is received, it is processed by the VM REXECD server in essentially the same manner as an REXEC command. The same user ID and password verification mechanisms used for REXEC commands are used for rsh-supplied commands as well. An attempt is made to log on to the VM user ID that is identical to the local login id used to issue the rsh command; the value used is usually that which is maintained in the LOGNAME variable on Unix-based systems. The only exception to this is when a special password value of ***guest** has been provided.

Because VM requires a password to be supplied when logging on to a user, a password must be available. The rsh and remsh commands however have no inherent provision for doing so. Moreover, there is no `rhosts` (or similar) file maintained by the VM host for remote user authentication purposes. To bypass these limitations, the *-l* flag of the rsh command is used to provide the VM logon password as part of the rsh command. For example, when the following rsh command is issued from an AIX system by the user `wellsjr`:

```
rsh gd3vm0 -l blues2me q cmslevel
```

the VM user ID WELLSJR will be logged on using the password BLUES2ME. The QUERY CMSLEVEL command is then executed on the host GD3VM0.

On other platforms, the rsh command may need to be specified using different flags. For example, from an OS/2 client, the above rsh command needs to be specified with both the *-l* and the *-u* flags:

```
rsh gd3vm0 -u welljr -l blues2me q cmslevel
```

If a user has no user ID on a remote VM system, rsh commands can still be issued, though in an anonymous manner. In this case, the supplied command will be processed by an REXEC agent virtual machine. To have an rsh command processed in this way, you need to specify a special value of ***guest** via the *-l* flag as shown below:

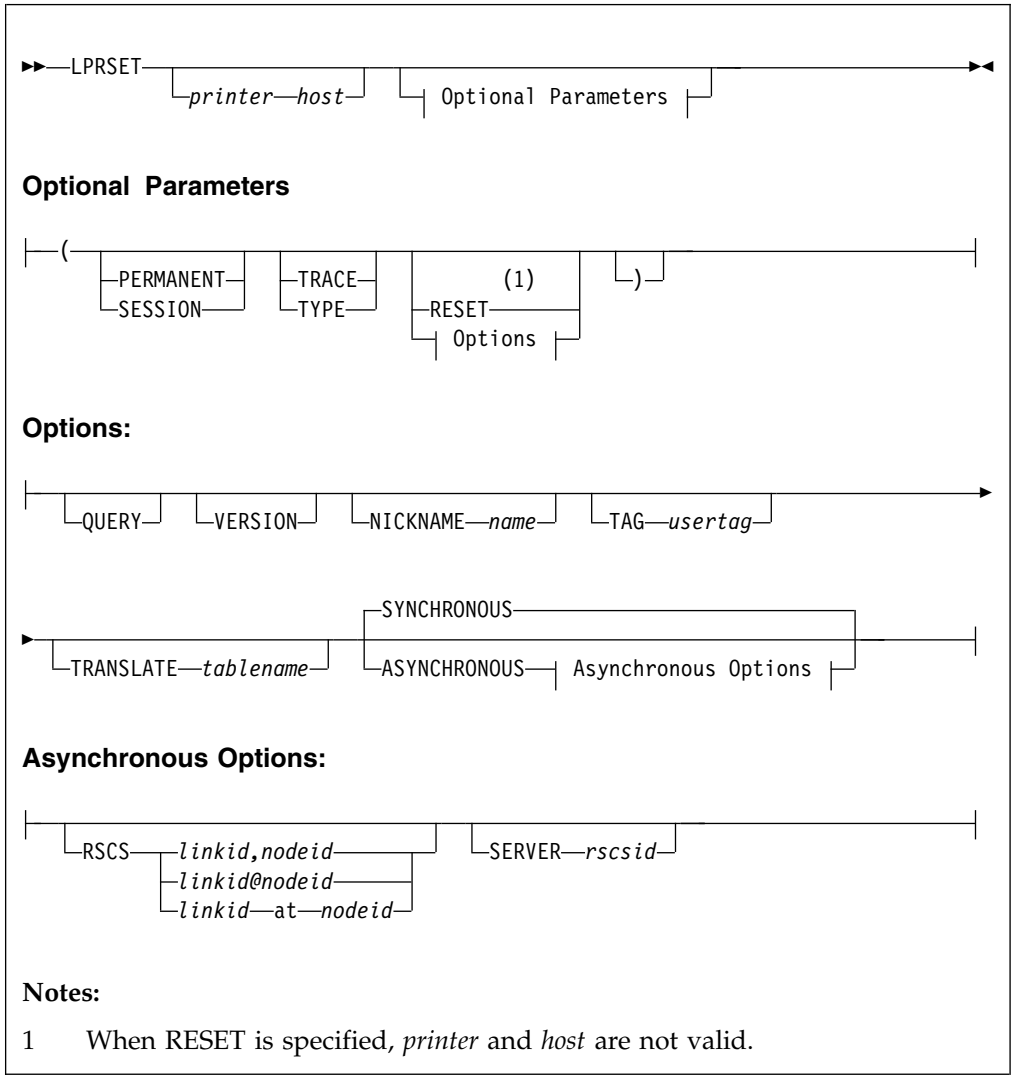
```
rsh gd3vm0 -l *guest q cmslevel
```

Chapter 11. Using Remote Printing

TCP/IP provides client support for remote printing. The remote printing application allows you to spool files remotely to a line printer daemon (LPD). The line printer client (LPR) sends the spooled file to a specified print server host and to a specified printer.

The following commands are described in this chapter:
- LPRSET
- LPR
- LPQ
- LPRM

LPRSET Command



Notes:

1 When RESET is specified, *printer* and *host* are not valid.

© Copyright IBM Corp. 1987, 2016

Purpose

Use the LPRSET command to define default printer and host values for the LPQ, LPR, and LPRM remote printing commands, or to set additional defaults for the LPR command.

Note: You can use the shortest available unique character sequence as a minimum abbreviation for an LPRSET parameter.

Operands

printer
Specifies the name of the printer to be used for the LPQ, LPR, and LPRM remote printing commands; *printer* is restricted to 255 characters.

host
Specifies the name or internet address (in dotted-decimal form) of the printer host to be used for the LPQ, LPR, and LPRM remote printing commands; *host* is restricted to 255 characters.

ASYNCHRONOUS
This option will send LPR requests to another server (RSCS) for later printing. As a result, this causes LPR operations to be deferred. When this option is in effect, subsequent LPR commands cause data to be queued to an intermediate print service machine (an RSCS server). As a result, interaction with a remote print daemon is handled by this service machine and ensures your file will be processed with no further action required by you. Messages and status information associated with a deferred transaction are returned to your console.

NICKNAME *name*
Specifies a nickname (defined in your CMS NAMES file) that identifies the printer and host to which files are directed for printing. Printer-specific options can also be defined using such an entry. See LPR Usage Note 3 on page 347 and "Usage Notes" on page 346 for more information about specifying values through nickname definitions and how nicknames and values can be used.

PERMANENT
Causes values to be maintained on a permanent basis (across separate LOGONs). By default, values are maintained only for the duration of the current initialization (IPL) of CMS. All values are affected when this option is used.

QUERY
Displays the current settings.

RESET
Causes **all** values to be re-initialized to null values. When this option is used in conjunction with the PERMANENT option, **all** values are reinitialized; when used with the SESSION option, only session-related values are reinitialized. If neither PERMANENT or SESSION is specified only current, in-storage values are affected.

For information about how to reset individual values, see Usage Note 2 on page 338.

RSCS *linkid*,*nodeid*
RSCS *linkid*@*nodeid*

RSCS *linkid* **AT** *nodeid*
Identifies the RSCS link that provides a connection to a target print daemon or device, where:

- *linkid* is the one- to eight-character link identifier of the RSCS link that provides the connection to the chosen print daemon or device.
- *nodeid* is the one- to eight-character node name of the remote VM system that provides the connection to a target print daemon or device.

When specific link and node values are not defined, "LPR" is used as the default link identifier (*linkid*) for non-PostScript files, while "LPRP" is used for PostScript files; the local node (as returned by the CMS IDENTIFY command) is used for *nodeid*.

Note: This option applies only to asynchronous LPR processing.

SERVER *rscsid*
Identifies an RSCS service virtual machine to which print data is to be spooled. By default, the RSCS server reported by the CMS IDENTIFY command is used.

Note: This option applies only to asynchronous LPR processing.

SESSION
Causes values to be defined for a session (generally, from LOGON to LOGOFF). By default, values are maintained only for the duration of the current initialization (IPL) of CMS. All values are affected when this option is used.

SYNCHRONOUS
Causes LPR operations to be processed immediately; this is the default. When this option is in effect, subsequent LPR commands are processed as immediate operations through the TCP/IP server and the chosen print daemon. Delays in command execution (caused by network congestion, for example) may be apparent, because synchronous processing is dependent upon the interaction between the TCP/IP service machine and the specified print daemon.

TAG *usertag*
Identifies a specific NAMES file tag from which printer and host destination information is to be retrieved. If such a tag is not identified, an attempt is made to retrieve printer and host values from a set of default tag definitions. For more information about how destination information is retrieved for a nickname entry, see LPR Usage Note 4 on page 347.

TRACE
Displays detailed command progress information.

TRANSLATE *tablename*
Identifies the file name of a translation table to be used for EBCDIC to ASCII data translation. See *TCP/IP Planning and Customization* for more information on creating and loading translation tables, as well as the "Using Translation Tables" chapter in this publication.

TYPE
Displays abbreviated command progress information.

VERSION
Displays program version information.

Usage Notes

1. Parameters established using the LPRSET command are maintained using CMS global variables. This allows these values to be shared by the various TCP/IP

remote printing commands and allows values to be retained (either temporarily or permanently) for subsequent use. For more information about CMS global variables, see the *z/VM: CMS Commands and Utilities Reference*.

Note: The *printer* and *host* values are recognized by all of the line printer commands; values set using options (such as NICKNAME and TAG) are recognized only by the LPR command.

2. For operands other than *printer* and *host*, values can be reinitialized on an individual basis. For such operands, a value specified as a single period (.) will cause the current value to be nullified. For the RSCS and SERVER options, this will cause an appropriate system default to be reinstated.
3. Parameters established with the LPRSET command can be overridden by using appropriate operands when LPQ, LPR and LPRM commands are issued.
4. Printer names may be case-sensitive, though this depends upon the host to which remote printing commands are directed. In many cases, the printer name you provide must match the printer definition used by the remote host. For example, on UNIX-like systems, `prt1` and `PRT1` can refer to different printers.
5. In most environments, the system defaults for asynchronous processing (that is, the default SERVER and RSCS *linkid* and *nodeid* values) should provide satisfactory results. Before setting non-default values for asynchronous processing, consult your RSCS operations support staff.

Examples

- To set the default printer and host as printer `LPTQ1` on host `prtsrv`, enter the following command:

  ```
  lprset LPTQ1 prtsrv
  ```

  The default values established with this LPRSET command will be lost if CMS is reinitialized (that is, re-IPL'd). To make this selection permanent, use the following command:

  ```
  lprset LPTQ1 prtsrv (perm
  ```

- To display the version of LPRSET currently in use, enter this command:

  ```
  lprset (ver
  ```

- To establish a nickname default so that values defined for the NAMES file entry `SIMPLPRT` will be used when LPR commands are processed, issue the following command:

  ```
  lprset (nick simplprt
  ```

- To set a nickname default so that values defined in your CMS NAMES file by the `CMPLXPRT` nickname will be used when LPR commands are processed, issue the following command:

  ```
  lprset (nick cmplxprt
  ```

  To ensure specific printer and host values defined by the :TSTPRINT tag entry (associated with the `CMPLXPRT` nickname) are used for LPR commands, use the TAG option in addition to NICKNAME option, as follows:

  ```
  lprset (nick cmplxprt tag tstprint
  ```

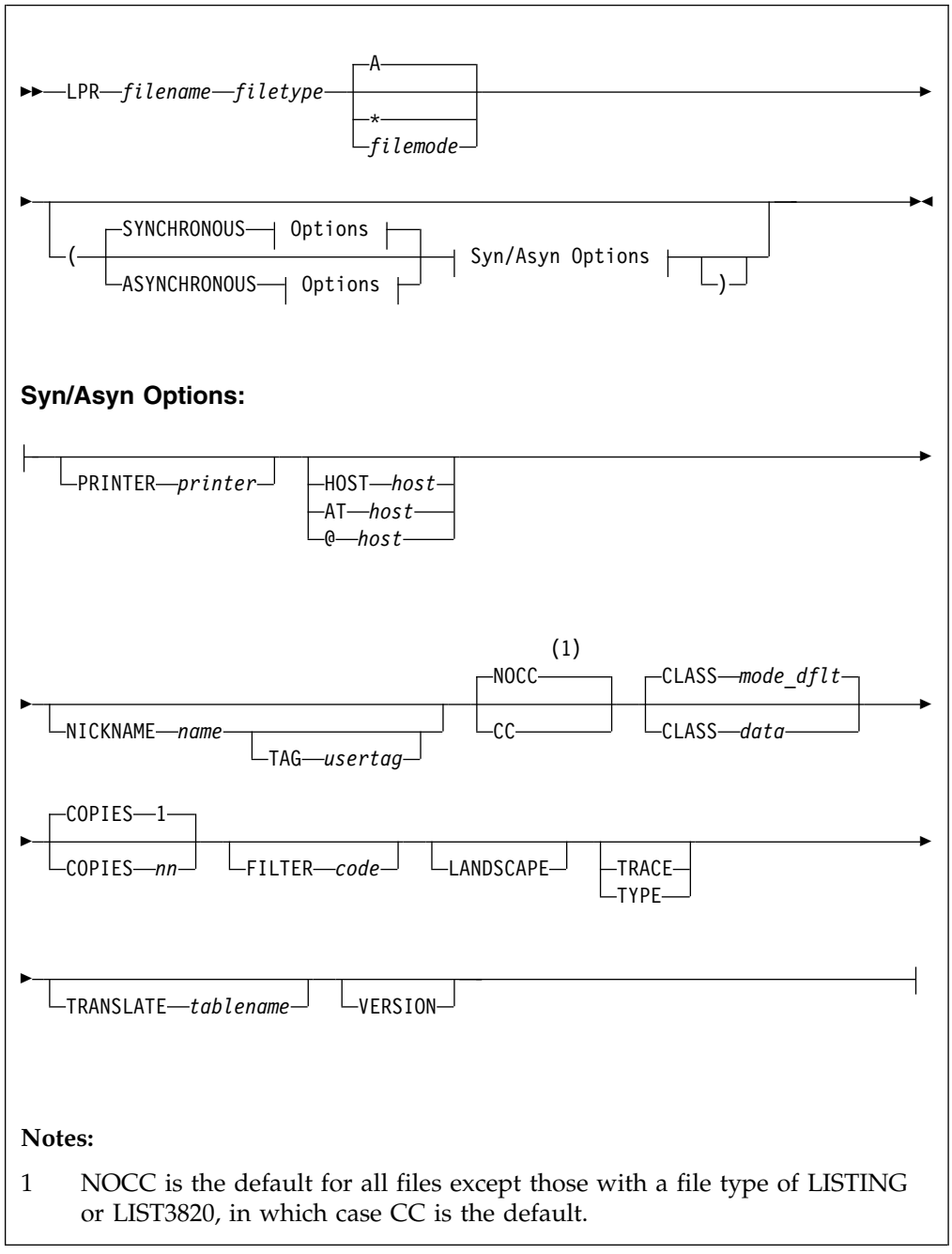
- To reset in-storage nickname and tag values so they will not be used for subsequent LPR commands, but still maintain current values for other parameters, use the following command:

  ```
  lprset (nick . tag .
  ```

- To establish the default translation table name as `MYXLATBL`, enter the following command:

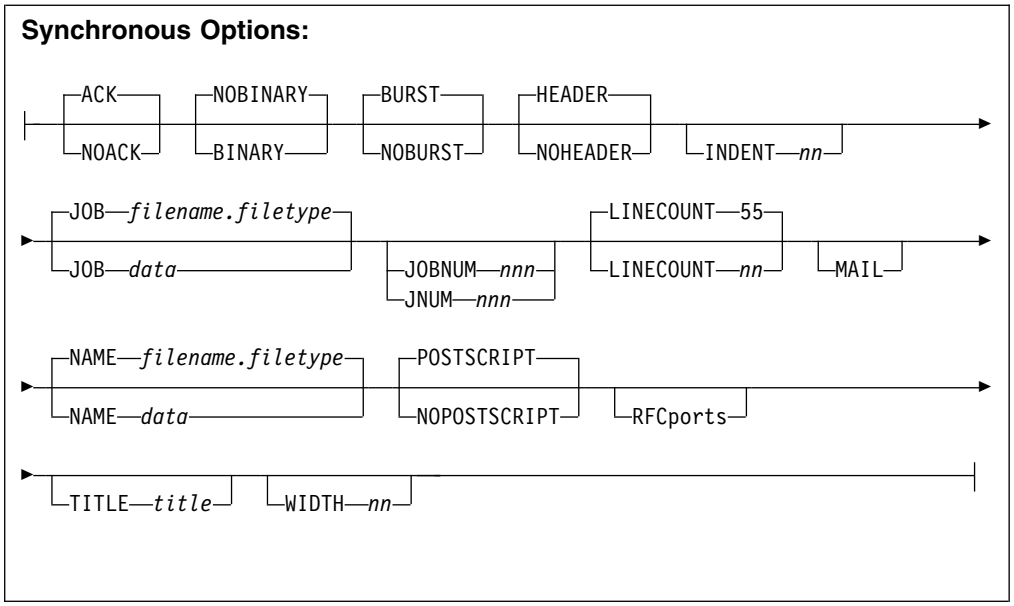
```
lprset (translate myxlatbl
```

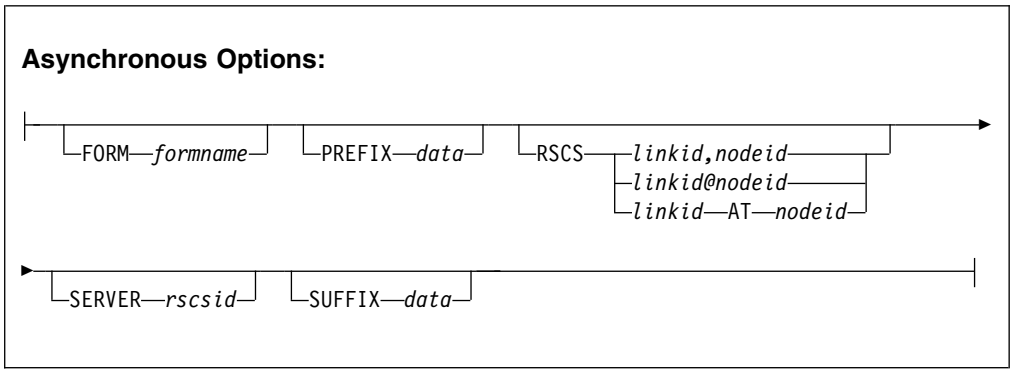
LPR Command



Notes:

1 NOCC is the default for all files except those with a file type of LISTING or LIST3820, in which case CC is the default.





Purpose

Use the LPR command to print files on remote printers.

Note: With some exceptions, you can use the shortest available unique character sequence as a minimum abbreviation for an LPR parameter. The exceptions are:

- **P** and **PR** are presumed to mean PRINTER
- **H** is presumed to mean HOST

Operands- Synchronous and Asynchronous

filename
Specifies the name of the file to print.

filetype
Specifies the type of the file to print.

filemode
Specifies the mode of the file to print. If *filemode* is specified as an asterisk (*), the first file found in the CMS search order is used. If *filemode* is omitted, a file mode of "A" is assumed. If a CMS file mode number is not specified, the default is 1.

Synchronous operations cannot be used to process files that have a file mode number of 3 or 4. For asynchronous operations, this restriction applies to only file mode number 3 files. See the *z/VM: CMS User's Guide* for information about file mode numbers and how they are used by CMS.

ASYNCHRONOUS
This option will send LPR requests to another server (RSCS) for later printing. As a result, this causes LPR operations to be deferred. When this option is in effect, subsequent LPR commands cause data to be queued to an intermediate print service machine (an RSCS server). As a result, interaction with a remote print daemon is handled by this service machine and ensures your file will be processed with no further action required by you. Messages and status information associated with a deferred transaction are returned to your console.

CC Causes the remote system to interpret the first character of each data line as ASA carriage control; this is the default for files with a file type of either LISTING or LIST3800.

NOCC
Prevents the remote system from interpreting the first character of each data line as ASA carriage control; this is the default for all files *except* those with a file type of either LISTING or LIST3800.

COPIES *nn*
Specifies the number of copies to be printed; the default is 1. Note that some remote servers may not recognize or honor requests to produce multiple copies of a file. For asynchronous operations, a maximum of 255 can be specified.

CLASS *data*
Specifies the print job classification to be used by the remote system when the print file is processed. By default, the host name (combined with the domain name, if available) defined in the TCPIP DATA file is supplied as *data* for synchronous operations; "A" is the asynchronous default.

For synchronous operations, *data* is restricted to 31 characters. For print requests directed to a UNIX-like lpd, *data* will be printed on the burst (or, banner) page.

For asynchronous operations, *data* specifies a spool file class. For such operations, *data* must be a single alphanumeric value (A-Z or 0-9) or an asterisk (*).

FILTER *code*
Specifies the type of processing to be performed against print file data by the remote system. The filter *code* value must be a single letter. Both uppercase and lowercase letters are accepted, but uppercase letters are converted to lowercase since only lowercase filter values are recognized by remote print daemons.

Filter codes provide carriage control. When a filter code is specified, LPR changes several of its conventions. When a filter code of **f**, **l**, or **r**, is specified, LPR does not paginate the processed file. Instead, it sends the data within the file as "plain" lines. The list that follows provides a description of available filter codes:

Code	Description
f	Print data as a sequence of lines
l	Print, passing through all control characters
p	Print with pagination

r Print, and interpret the first column as FORTRAN (ASA) carriage control. The following characters are interpreted as indicated:

Character	Printing Action
Space	Start a new line of output
+	Overprint this line on the previously printed line
-	Produce two blank lines, then begin a new line of output
0	Produce one blank line, then begin a new line of output
1	Start new page, and begin printing at first line

For filter codes **c**, **d**, **g**, **n**, **t**, or **v**, LPR transmits the data as a byte stream (as though the BINARY option has been specified).

HOST *host*
AT *host*
@ *host*
Specifies the name or internet address (in dotted-decimal form) of the printer host where the file is to be printed; *host* is restricted to 255 characters. **H** is accepted as the minimum abbreviation for the HOST keyword.

LANDSCAPE
For synchronous operations, this option causes PostScript information to be added to the print file so that it will be printed by the remote system in landscape (rotated) format — provided the remote printer can process PostScript.

For asynchronous operations, this option causes a form name of "LA" to be used when the file is processed. When used in this context, this option will override any FORM value specified as part of a CMS NAMES file entry.

NICKNAME *name*
Specifies a nickname (defined in your CMS NAMES file) that identifies the printer and host to which files are directed for printing. Printer-specific options can also be defined using such an entry. See Usage Note 3 on page 347 and "Usage Notes" on page 346 for more information about specifying values through nickname definitions and how nicknames and values can be used.

PRINTER *printer*
Specifies the name of the printer on which the file is to be printed; *printer* is restricted to 255 characters. **P** is accepted as the minimum abbreviation for this option.

SYNCHRONOUS
Causes LPR operations to be processed immediately; this is the default. When this option is in effect, subsequent LPR commands are processed as immediate operations through the TCP/IP server and the chosen print daemon. Delays in command execution (caused by network congestion, for example) may be apparent, because synchronous processing is dependent upon the interaction between the TCP/IP service machine and the specified print daemon.

TAG *usertag*
Identifies a specific NAMES file tag from which printer and host destination information is to be retrieved. If such a tag is not identified, an attempt is made to retrieve printer and host values from a set of default tag definitions. For more information about how destination information is retrieved for a nickname entry, see Usage Note 4 on page 347.

TRACE
Displays detailed command progress information. For synchronous operations, information about the interaction with the remote printer is also displayed.

TRANSLATE *tablename*
Identifies the file name of a translation table file to be used for EBCDIC to ASCII data translation; the file type for this file must be TCPXLBIN. The first `tablename` TCPXLBIN file found in the CMS search order is used. If this parameter is not specified, a default translation table is used (if one exists), which is defined by either a CMS NAMES file nickname entry or a CMS global variable; if neither exist, data translation is then performed as follows:

- For synchronous processing, LPR searches for and uses the LPR TCPXLBIN file first, and then the STANDARD TCPXLBIN file. If neither is found, an internal translation table is used.
- For asynchronous processing, default data translation processing is performed by the RSCS server to which files are directed, based on the configuration of that server.

For more information about creating and loading translation tables, see *z/VM: TCP/IP Planning and Customization*.

If your use of LPR requires specific translations to be performed, see Chapter 15, "Using Translation Tables," on page 437.

TYPE
Displays abbreviated command progress information.

VERSION
Displays program version information.

Operands - Synchronous

ACK
Requests the remote printer host to send an acknowledgment to the LPR command when the connection to that host is closed; this is the default.

NOACK
Requests the remote printer host to not return a receipt acknowledgment to the LPR command. This option was previously required if the receiving system was AIX; however, its use is no longer necessary.

BINARY
Causes print data to be sent without translation and without any indication of record boundaries. For example, use this option if the file is already in ASCII. At times a filter may also be required to achieve appropriate results; see the description of the FILTER operand in "Operands- Synchronous and Asynchronous" on page 340 for more information about filter use. The BINARY option implies that the file to be printed is not PostScript.

NOBINARY
Causes print data to be converted from EBCDIC to ASCII when it is sent to the remote system; this is the default.

BURST
Causes a burst page to be printed on the remote printer; this is the default. This option controls whether burst (or, banner) page information is sent to the server. If a VM LPD server does not receive burst page information, it provides the best information available to a spool device, as CMS does not have an option to omit the burst page for print files.

NOBURST
Prevents a burst page from being printed on the remote printer.

HEADER
Causes a page header to be inserted by the LPR client at the beginning of every printed page — if the NOCC and NOBINARY options are in effect. HEADER is the default. To instead cause the remote printer to insert page headers, use the FILTER **p** and NOHEADER options.

NOHEADER
Prevents the LPR client from inserting page headers.

INDENT *nn*
Specifies the number of columns the remote printer indents output when the FILTER **f** or FILTER **l** options are in effect.

JOB *data*
Specifies a print job description, name, or other job information to be used by the remote system; a maximum of 99 characters can be specified. By default, the string "*filename.filetype*" is used for *data*.

For a UNIX-like lpd, this option causes job information to be printed on the banner page; for a VM LPD server, it can be used to pass additional job information for use by that LPD server.

JOBNUM *nnn*
JNUM *nnn*
Specifies the job number used to construct the file names of protocol-specific files sent to the remote print server. The provided value must be a three-digit, numeric string. If a specific string is not specified using this option, a three-digit random number is used.

This option can be used to reduce the likelihood of problems caused by duplicate file names when many LPR jobs are initiated. To be effective toward this end, the files to be printed must be processed by a single VM user ID; as LPR commands are issued for each file, sequential job numbers should be specified using the JOBNUM option. However, collisions with jobs run by other virtual machines are still possible; a collision occurs when one of the print jobs fails.

LINECOUNT *nn*
Specifies the number of lines to be printed before a new heading is printed; the default is 55. This parameter has meaning only for files for which the CC option is not in effect (either explicitly or implicitly). A value of zero (0) can be used to prevent page ejects and page headers from being inserted in the print file.

MAIL
Causes mail to be sent to you when the printing operation ends that informs you about the success or failure of the print job. This option may not recognized by all remote printing servers.

NAME *data*
Specifies job name information to be provided by the remote system in response to a remote printer query (that is, an LPQ command). The supplied data is restricted to 131 characters. By default, the string "*filename.filetype*" is used for *data*. This option is not recognized by all remote printing servers.

POSTSCRIPT
Causes header information to be added to the print file so that it will be recognized as PostScript by the remote printer.

NOPOSTSCRIPT
Prevents a file from being recognized as PostScript.

RFCports
Enforces the use of RFC-compliant printer source ports when print requests are processed. Some printer host machines require the source port to be within either the 721-731 port range, or the alternative range of all "well-known" ports (1-1023).

Note: By default, TCP/IP for z/VM restricts general use of well-known ports. To use the RFCPORTS operand, you may need to contact your local TCP/IP administrator to obtain authorization to use such ports.

TITLE *title*
Specifies the title assigned to a file printed with the FILTER **p** option. A maximum of 79 characters can be specified for *title*.

WIDTH *nn*
Specifies the line width to be used for a file printed with the FILTER **f** or FILTER **l** option. The value you provide may be decreased by the remote printer host.

Operands - Asynchronous

FORM *formname*
Specifies a spool file form name (used by RSCS) to control how data is printed at the destination printer. For PostScript data, *formname* can be specified as a character string of the form *OrFnFsLs*, to specify a default orientation, font name, font size, and additional leading size to be used by the destination printer. For *OrFnFsLs* the following can be specified:

Or File Orientation

- **PO** Portrait (default)
- **LA** Landscape

Fn Font

- **CB** Courier Bold
- **CI** Courier Oblique
- **CP** Courier (default)
- **CX** Courier BoldOblique
- **HB** Helvetica Bold
- **HP** Helvetica
- **HI** Helvetica Oblique
- **HX** Helvetica BoldOblique
- **SP** Symbol
- **TB** Times Bold
- **TI** Times Italic
- **TP** Times Roman
- **TX** Times BoldItalic

Fs Font sizes, 04-99. The default is 11 for portrait (PO) and 10 for landscape (LA).

Ls Additional leading size, 0.0-9.9 added to font size to give leading, specified as 00 thru 99. The portrait (PO) default is 09; that for landscape (LA) is 12.

Note:

1. The font specified via *OrFnFsLs* must be installed and used by the destination printer for correct results to be achieved.
2. For ASCII PostScript files, the default form name used is "P+ASCII"; for EBCDIC PostScript files, the default is "P+SCRIPT". There is no default for non-PostScript files.
3. FORM cannot be used with the LANDSCAPE option.
4. The spool file form name (*formname*) is translated to uppercase prior to use.

PREFIX *data*
Specifies a data string to be passed to the remote printer device by the RSCS server; up to 500 characters can be specified. Prefix *data* is translated to uppercase and is inserted at the beginning of the data file by RSCS, and might be used to affect printer settings; for example, to select a paper tray.

RSCS *linkid*,*nodeid*
RSCS *linkid*@*nodeid*
RSCS *linkid* **AT** *nodeid*
Identifies the RSCS link that provides a connection to a target print daemon or device, where:

- *linkid* is the one- to eight-character link identifier of the RSCS link that provides the connection to the chosen print daemon or device.
- *nodeid* is the one- to eight-character node name of the remote VM system that provides the connection to a target chosen print daemon or device.

When specific link and node values are not defined, "LPR" is used as the default link identifier (*linkid*) for non-PostScript files, while "LPRP" is used for PostScript files; the local node (as returned by the CMS IDENTIFY command) is used for *nodeid*.

SERVER *rscsid*
Identifies an RSCS service virtual machine to which print data is to be spooled. By default, the RSCS server reported by the CMS IDENTIFY command is used.

SUFFIX *data*
Specifies a data string to be passed to the remote printer device by the RSCS server; up to 500 characters can be specified. Suffix *data* is translated to uppercase and is inserted at the end of the data file by RSCS. A suffix data string might be used to affect printer settings; for example, to reset the printer state.

Usage Notes

1. When LPR commands are issued and certain operands are omitted, LPR will attempt to use values defined by CMS global variables for the TCPIP group. Operands to which this applies are:
   - *printer* and *host*
   - RSCS *linkid* and *nodeid*
   - Mode of operation (SYNCHRONOUS or ASYNCHRONOUS)
   - NICKNAME *name* and TAG *usertag*
   - TRANSLATE *tablename*

   Values for these operands can be established and changed using the LPRSET command.

2. LPR command operands are selected, in order, from the following sources:
   a. the LPR command itself
   b. CMS NAMES file entries
   c. CMS global variables

   Values from different sources may be combined and used for a single LPR command.
3. CMS NAMES file tags recognized by the LPR command and the options to which they correspond follow:

   Tag Corresponding Option(s)
   :CSADDR
   PRINTER and HOST
   :FILTER
   FILTER
   :LPRADDR
   PRINTER and HOST
   :NODE
   HOST
   :TCPADDR
   PRINTER and HOST
   :USERID
   PRINTER
   :TRANSLATE
   TRANSLATE

   :LINKID
   RSCS
   :NODEID
   RSCS
   :PREFIX
   PREFIX
   :SERVER
   SERVER
   :SUFFIX
   SUFFIX
   :FORM
   FORM

   Tags listed in the first group are supported for both synchronous and asynchronous operations; those listed in the second group are applicable only to asynchronous use. For all tags, values should be specified in the same manner as for their corresponding options.
4. Print destination information (that is, the combination of a printer and host name) is obtained for NAMES file entries, in order, from the following tags:
   a. a user-specified tag, as identified by the TAG option
   b. an "address" tag, if the TAG option is not specified. Address tags are checked, when present, in this order:
      1) :LPRADDR
      2) :TCPADDR
      3) :CSADDR
   c. the :USERID and :NODE tags, if no information is defined by any of the previously listed tags. In the context of using TCP/IP remote printing commands, these tags are presumed to provide printer and host names, respectively, instead of conventional user ID and node ID information.

Printer and host values for user-specific tags and the address tags previously listed can be specified using one of these formats:
- *linkid nodeid*
- *linkid,nodeid*
- *linkid@nodeid*
- *linkid* AT *nodeid*

5. Printer names may be case-sensitive, though this depends upon the host to which remote printing commands are directed. In many cases, the printer name you provide must match the printer definition used by the remote host. For example, on UNIX-like systems, `prt1` and `PRT1` can refer to different printers.
6. For certain operands, values can be specified as quoted strings, so that the content between the string delimiters — either single (') or double (") quotation marks — is preserved. Operands for which quoted values are accepted are: PRINTER, HOST, CLASS, JOB, NAME, and TITLE.
7. The LPR command can be used to print PostScript documents. When a file is processed by LPR, the file is checked to determine whether it is a PostScript file. If it is, additional checks are performed (for synchronous operations only) to verify that compatible options have been provided. If you want to override these checks when you print a PostScript file, use the NOPOSTSCRIPT option.

   Remote hosts usually examine the first few characters of a file to determine if that file is PostScript; this is usually indicated by the presence of the character string "`%!`" in the first line of the file. For synchronous operations, the POSTSCRIPT option can be used to ensure a file is recognized as PostScript.
8. Carriage control is interpreted line by line. A file can mix ASA and machine carriage control. Interpretation is done by converting the controls to the appropriate ASCII sequences before the file is sent to the remote system. Lines that have invalid carriage control are not printed.

   When a file is printed without carriage control, LPR adds a heading line to each page of output that shows the name of the printed file, the name of the system on which the LPR command originated, and a page number. You can specify the number of lines per page (not counting the three heading lines) with the LINECOUNT option.
9. The logical record length (LRECL) of files that can be processed is restricted for both synchronous and asynchronous operations.
   - For synchronous operations, LPR processes files using OS file routines. Thus for fixed-format files, the maximum LRECL is 32760; for variable-format files, the maximum LRECL is 32756.
   - For asynchronous operations, files are processed using RSCS services. For non-PostScript (or, "flat") files, only those with an LRECL of 1280 or less can be processed in this manner; there is no restriction for PostScript files.
10. Some LPDs require the FILTER **l** option — in addition to the BINARY option — to ensure the data file is not changed during BINARY transfers. With such implementations, the receiving LPD converts the incoming line-end character of X'0A' to two characters, X'0D0A', if the FILTER **l** option is omitted. (This situation has been observed when files are processed by an LPD on some DOS and OS/2 based systems; however, this behavior may not occur in all such environments.)
11. Remote printer hosts determine whether sufficient space is available to receive a given file before it is processed. If sufficient space is not available, the file is discarded and the connection is terminated with a message that identifies this error condition.

12. In most environments, the system defaults for asynchronous processing (that is, the default SERVER and RSCS *linkid* and *nodeid* values) should provide satisfactory results. Before setting non-default values for asynchronous processing, consult your RSCS operations support staff.
13. For most operations, LPR issues messages only if errors are encountered. To monitor the progress of an LPR command or to obtain information for diagnostic purposes, use the TYPE or TRACE options.

Examples

Examples for General Printing:

- To print the file TEST LISTING on printer LPTQ1 on the host prtsrv, enter the following command:

  ```
  lpr test listing (printer LPTQ1 host prtsrv
  ```

  Because the file type is LISTING, the first character of each line is interpreted as carriage control. To prevent this, use the NOCC option as shown in the following command:

  ```
  lpr test listing (printer LPTQ1 at prtsrv nocc
  ```

- If an LPRSET command was previously issued to set the printer and host defaults to LPTQ1 and prtsrv (for example, LPRSET LPTQ1 prtsrv), the following command has the same effect as the second command shown in the previous LPR example:

  ```
  lpr test listing (nocc
  ```

  Because the lines in a listing file may be wider than a page, you may want to print the listing in "landscape" mode. The next example includes the LANDSCAPE option to print the TEST LISTING file in this mode:

  ```
  lpr test listing (landscape
  ```

- To print a source program (CSPROG1 FORTRAN) so that 57 lines are printed per page, enter the following command:

  ```
  lpr csprog1 fortran (linecount 57
  ```

- To print a file and have it processed by an RSCS server named RSCSTST, enter the following command:

  ```
  lpr demotest schedule (async server rscstst printer prt01@prtsys1
  ```

  In the above example, the file DEMOTEST SCHEDULE is processed by the RSCSTST server, and sent to the printer prt01 defined for the local host PRTSYS1.

- To print a file and have data translation performed based on the translation table named MYXLATBL, enter the following command:

  ```
  lpr trantest datafile (printer LPTQ1 host prtsrv translate myxlatbl
  ```

  In the above example, file TRANTEST DATAFILE is sent to the printer LPTQ1 defined for the host prtsrv; the MYXLATBL TCPXLBIN table file is used to perform data translation.

- To print the file TEST LISTING on printer LPTQ1 on the host prtsrv, enter the following command:

  ```
  lpr test listing (printer LPTQ1 host prtsrv
  ```

  Because the file type is LISTING, the first character of each line is interpreted as carriage control. To prevent this, use the NOCC option as shown in the following command:

  ```
  lpr test listing (printer LPTQ1 at prtsrv nocc
  ```

- If an LPRSET command was previously issued to set the printer and host defaults to LPTQ1 and prtsrv (for example, LPRSET LPTQ1 prtsrv), the following command has the same effect as the second command shown in the previous LPR example:

  ```
  lpr test listing (nocc
  ```

  Because the lines in a listing file may be wider than a page, you may want to print the listing in "landscape" mode. The next example includes the LANDSCAPE option to print the TEST LISTING file in this mode:

  ```
  lpr test listing (landscape
  ```

- To print a source program (CSPROG1 FORTRAN) so that 57 lines are printed per page, enter the following command:

  ```
  lpr csprog1 fortran (linecount 57
  ```

- To print a file and have it processed by an RSCS server named RSCSTST, enter the following command:

  ```
  lpr demotest schedule (async server rscstst printer prt01@prtsys1
  ```

  In the above example, the file DEMOTEST SCHEDULE is processed by the RSCSTST server, and sent to the printer prt01 defined for the local host PRTSYS1.

- To print a file and have data translation performed based on the translation table named MYXLATBL, enter the following command:

  ```
  lpr trantest datafile (printer LPTQ1 host prtsrv translate myxlatbl
  ```

  In the above example, file TRANTEST DATAFILE is sent to the printer LPTQ1 defined for the host prtsrv; the MYXLATBL TCPXLBIN table file is used to perform data translation.

Examples for Printing Using Nicknames: The examples that follow illustrate some typical NAMES file entries that might be constructed for use with the LPRSET or LPR commands. For detailed information about defining and using nicknames, see Usage Notes 2 on page 347, 3 on page 347 and 4 on page 347.

- The following example shows a simple NAMES file entry that defines only a printer and host name.

  ```
  :nick.SIMPLPRT :userid.lpt1     :node.oddjob.endicott.ibm.com
  ```

  The command:

  ```
  lpr profile exec (nick simplprt
  ```

  will print file PROFILE EXEC on the oddjob.endicott.ibm.com host printer lpt1.

- In the next example, additional tag entries are included as part of a nickname entry.

  ```
  :nick.ADVPRT1  :userid.nullprt  :node.nowhere.endicott.ibm.com
      :myprint.prt1@paradox.endicott.ibm.com
      :lpraddr.LPT2@monolith.endicott.ibm.com
  ```

  With the above entry, two different destinations can be used for printing, based on the options provided with an LPR command.

  When following command is issued:

  ```
  lpr deptg79 report (nick advprt1
  ```

  the printer and host name defined by the :LPRADDR tag are used (LPT2 and monolith.endicott.ibm.com). The :LPRADDR tag definition overrides the printer and host information defined by the :USERID and :NODE tags. This would also occur if LPT2 and monolith.endicott.ibm.com were defined using either a :TCPADDR or :CSADDR tag. For more information about how printer and host information is obtained for nickname entries, see Usage Note 4 on page 347.

If instead, the following command is used:

```
lpr deptg79 report (nick advprt1 tag myprint
```

the printer and host name defined by the :MYPRINT tag are used (prt1 and paradox.endicott.ibm.com). Again, any printer and host information defined by the :USERID and :NODE tags is ignored because the TAG option is specified.

For asynchronous processing of the DEPTG79 REPORT file, the following command could be used:

```
lpr deptg79 report (asynch nick advprt1
```

For this command, the DEPTG79 REPORT file would first be passed to the RSCS server of the local node (the default), and then would be sent to the prt1 printer of the paradox.endicott.ibm.com host. Because no RSCS link identifier was provided, a default link of either LPR or LPRP would be used.

- This last example illustrates how various tags might be defined and then used for synchronous and asynchronous remote printing requests.

```
:nick.CMPLXPRT
    :tstprint.lpt0@rocketman.endicott.ibm.com
    :tcpaddr.PRTQ1@monolith.endicott.ibm.com
    :server.rscstst
    :linkid.lprtst
    :nodeid.GDLVME
* Prefix string to turn duplexing OFF; the PostScript command
* string is:  %!PS-AdobeCRLF statusdict begin false setduplexmode
*
* This string should be one contiguous line; it spans multiple
* lines here only to meet formatting requirements.

    :prefix.252150532D41646F62650D0A737461747573646963742062656 76
            96E2066616C736520736574647 5706C65786D6F646520656E640D0A

 * Suffix string to turn duplexing (back) ON; the PostScript command
 * string is:  %!PS-AdobeCRLF statusdict begin true setduplexmode
 *
 * This string should be one contiguous line; it spans multiple
 * lines here only to meet formatting requirements.

    :suffix.252150532D41646F62650D0A737461747573646963742062
            6567696E20074727565207365746475706C65786D6F646520656E640D0A
    :translate.MYXLATBL
```

If the nickname CMPLXPRT is specified in conjunction with the SYNCHRONOUS option on an LPR command, only values defined by the following tags will be used:

– :TSTPRINT is used if TAG TSTPRINT is specified
– :TCPADDR is used if the TAG option is not specified.
– :TRANSLATE is used if the TRANSLATE option is not specified.

If this same nickname is specified, but with the ASYNCHRONOUS option:

– :TSTPRINT is used if TAG TSTPRINT is specified
– :TCPADDR is used if the TAG option is not specified
– :TRANSLATE is used if the TRANSLATE option is not specified.

In addition, the values defined by all of the remaining tags shown will be used for processing — assuming no options are used that override them. For example, if the following command is issued:

```
lpr weather report (asynch nick cmplxprt tag tstprint
```

the WEATHER REPORT file would first be passed to the RSCSTST RSCS server, which then sends it to the RSCS node GDLVME. From GDLVME it is printed on printer lpt0 at host rocketman.endicott.ibm.com, using the RSCS link LPRTST.

Because the :PREFIX and :SUFFIX tags define values, this data will also be passed to the RSCSTST server. Also, since the TRANSLATE option was not specified, data translation will be performed using the translation table named `MYXLATBL`, as defined by the :TRANSLATE tag.

If the `TAG tstprint` option were omitted in the above command, the destination printer and host would instead be `PRTQ1` and `monolith.endicott.ibm.com`.

Controlling LPSERVE from other Systems

When you use the LPR command to communicate with LPSERVE, you can provide additional information for RSCS services. This can be done by passing the desired information with the client LPR command options.

For example, the VM LPR CLASS option or the UNIX lpr -C option is used by lpd in UNIX to specify a classification that is printed on the burst page. The VM CLASS or UNIX -C option, when sent to a VM LPD server, is used to alter the VM spooled file's class.

The VM LPR JOB option or the UNIX lpr -J option is used by lpd in UNIX to specify what is printed in this field of the burst page. The VM JOB or UNIX -J option, when used with a VM LPD server, specifies additional information to the VM server such as the distribution code, priority, password, or an alternative user ID or destination. Within the JOB or -J option, additional options can be used to specify these additional job parameters.

The additional JOB options for RSCS services are:
- FOR, PASS, DEST, IDENTIFIER, PRIORITY, and OTHERS

If you use more than one option, separate each by a comma with no intervening spaces between options. If the OTHERS option is used, it must be the last option. You can enter keywords on the LPR command line in any combination of upper and lower case.

Option Description

FOR=*user_id*
: Specifies a user ID other than the sending user ID for which the job is to be spooled. The default is the sender's user ID.

PASS=*password*
: Specifies the password for *user_id*. The default is no password. This option is required only if the RACF option has been specified for the designated service.

DEST=*destination*
: Specifies a RSCS destination node. The default is the node on which LPSERVE is running.

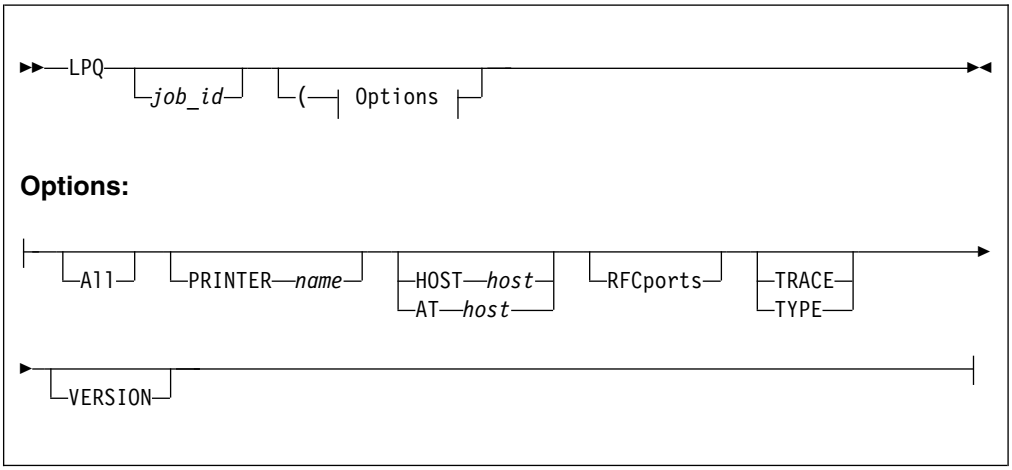
IDENTIFIER=*operand*
: Specifies the second TAG operand, which can be SYSTEM, JOB, a virtual machine user ID, or a workstation node ID. The default is SYSTEM.

PRIORITY=*nn*
: Specifies the transmission priority. The default is 50.

OTHERS=*option_name*
: Specifies additional options for the destination RSCS. This option applies only if the specified service has the TAG option specified and is used to supply additional TAG information. This can, for example, be used to

designate forms (for an MVS Network Job Entry (NJE) system) or controls for an IBM 3800 printer. The rest of the job data is used to tag the spooled file. The default is no additional options.

LPQ Command



Purpose

Use the LPQ command to list the printer queue on a remote printer.

Note: You can use the shortest unique sequence as the minimum abbreviation for an LPQ parameter.

Operands

job_id
Specifies either a user ID (must not start with a number), or a job number on the remote printer queue. If you do not specify *job_id* with the LPQ command, all jobs in the remote printer queue are listed.

All
Displays for all printers a report that shows print job information.

PRINTER *name*
Specifies the name of the printer for which printer queue information is to be obtained.

HOST *host*
AT *host*
Specifies the name or internet address of the printer host. AT is accepted as a synonym for HOST.

RFCports
Enforces the use of RFC-compliant printer source ports when printer queues are queried. Some printer host machines require the source port to be within either the 721-731 port range, or the alternative range of all "well-known" ports (1-1023).

Note: By default, TCP/IP for z/VM restricts general use of well-known ports. To use the RFCPORTS operand, you may need to contact your local TCP/IP administrator to obtain authorization to use such ports.

TRACE
Displays detailed information about the interaction with the remote printer.

TYPE
Displays the progress of the command.

VERSION
Displays the version of the program.

Usage Notes

1. The LPQ command allows you to query the printer queues on remote systems that support this activity.
2. If a printer or host name is not provided on the LPQ command, LPQ will use the GLOBALV variables PRINTER and PRTHOST, respectively, from the TCPIP group. These variables can be set by a program or by the LPRSET command in order to provide defaults for these values.
3. Printer names can be case sensitive. The printer you provide must match the remote host's definition for that printer. For example, on UNIX systems, `psnt` and `PSNT` can refer to different printers.
4. A user name, when provided as a *job_id*, can be case sensitive. For example, `smith` and `SMITH` may refer to different users on the same host.
5. Some systems will not respond with job information, if you use a job number for a job that was not produced by the querying system.

Examples

- To query the printer `psnt` on host system `test1`, enter the following command:

  ```
  LPQ (PRINTER psnt HOST test1
  ```

  This command displays a short listing of the jobs that are queued for the `psnt` printer.
- If the LPRSET command was previously issued, (LPRSET *psnt test1)*, the following LPQ command has the same effect as issuing the command in the first LPQ example:

  ```
  LPQ
  ```
- To get a long listing, enter the following command:

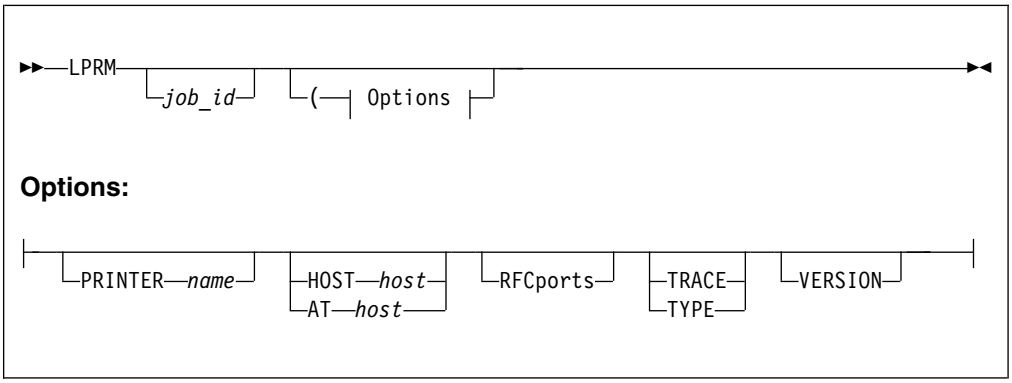
  ```
  LPQ (PRINTER psnt HOST test1 ALL
  ```

  This command prints a long listing of the jobs queued, including the name of the host that created the jobs.
- To list the jobs for a user named `smith`, enter the following command:

  ```
  LPQ smith (PRINTER psnt HOST test1
  ```
- If you want only the information about job 123, enter the following command:

  ```
  LPQ 123 (PRINTER psnt HOST test1
  ```

LPRM Command



Purpose

Use the LPRM command to remove a job from the printer queue on a remote host.

Note: You can use the shortest unique sequence as the minimum abbreviation for an LPRM parameter.

Operands

job_id
Specifies either a user ID (must not start with a digit, or a job number in the remote printer queue. If you do not specify *job_id* with the LPRM command, your currently active job is removed.

PRINTER *name*
Specifies the name of the printer with which the job is associated.

HOST *host*
AT *host*
Specifies the name or internet address of the printer's host. AT is accepted as a synonym for HOST.

RFCports
Enforces the use of RFC-compliant printer source ports when a job is removed from a printer queue. Some printer host machines require the source port to be within either the 721-731 port range, or the alternative range of all "well-known" ports (1-1023).

Note: By default, TCP/IP for z/VM restricts general use of well-known ports. To use the RFCPORTS operand, you may need to contact your local TCP/IP administrator to obtain authorization to use such ports.

TRACE
Turns on the trace details for the interaction with the remote printer.

TYPE
Displays the progress of the command.

VERSION
Displays the version of the program.

Usage Notes

1. The LPRM command allows you to remove jobs from printer queues on remote systems that support this capability.
2. If a printer or host name is not provided on the LPRM command, LPRM will use the GLOBALV variables PRINTER and PRTHOST, respectively, from the TCPIP group. These variables can be set by a program or by the LPRSET command in order to provide defaults for these values.
3. Removing the currently active job can be sensitive to timing. If you have two jobs printing, and you use the LPRM command without the *job_id* parameter, the first job may finish, and you may inadvertently remove the second job.

Examples

- To cancel job number 123 on the printer `psnt` on the local system `test1`, enter the following command:

  ```
  LPRM 123 (PRINTER psnt HOST test1
  ```

  If you printed the job, it is removed. If the job is currently active, it is stopped.
- If the LPRSET command *LPRSET psnt test1* was previously issued, entering the following LPRM command has the same effect as issuing the command in the first LPRM example:

  ```
  LPRM 123
  ```
- To cancel the currently active job, enter the following command:

  ```
  LPRM (PRINTER psnt HOST test1
  ```

Chapter 12. Using GDDMXD/VM with the X Window System

TCP/IP provides the GDDMXD/VM interface for the IBM Graphical Data Display Manager/VM (GDDM*), which allows graphics to be displayed on workstations that support the X Window System.

This chapter describes GDDMXD/VM and the GDDMXD command. This chapter also describes how to use GDDMXD/VM user-specified options and keyboard functions. Problem determination information associated with GDDMXD/VM is also described in this chapter.

Overview of GDDMXD/VM

When GDDMXD/VM is installed and activated, the data stream created by GDDM is translated to the X Window System protocol and transmitted by TCP/IP to the X Window System server for the display. If GDDMXD/VM is installed and not activated, or has been made inactive, GDDM transmits data to its supported display device as if GDDMXD/VM was not present.

GDDM Application Limitations

GDDMXD/VM does not support the following types of applications:

- Multiple instances of GDDM/VM
- Opening multiple display devices at one time
- Operator windows

GDDM Display Limitations

GDDMXD/VM appears as an IBM 3179G Color Graphics Display Station device model to GDDM. When the HostRast option is active, the device model is an IBM 3279. The IBM 3179G model is used regardless of the display device nickname presented by the application.

The GDDMXD/VM IBM 3179G device model contains the following features:

Feature
Description

Blink Attribute
Ignores the blinking character attribute.

Detectable Fields
Ignores the selector pen detectable fields.

Character Display
Displays characters with an EBCDIC value of less than X'40' as blanks.

Graphics Cursor
The X Window System pointer device cursor for the GDDMXD window is a crosshair pattern and moves as the pointer device is moved.

Alphanumeric Cursor
The alphanumeric pointer device is an open arrow when the keyboard is unlocked, or an “X” when the keyboard is locked. The cursor can be repositioned by moving the pointer to the desired character location and pressing a pointing device button.

© Copyright IBM Corp. 1987, 2016

Pixel Spacing
When the HostRast option is not active, the vertical and horizontal pixel spacing of the actual display device that is obtained from the X Window System is supplied to GDDM. When the HostRast option is active, the pixel spacing of an IBM 3279 Color Display Station is supplied to GDDM.

Appearance
For all programmed symbol and image data that is received from GDDM, each GDDM pixel is mapped to one X Window System display pixel, which causes a different appearance from the same data stream displayed on an IBM 3179G Color Graphics Display Station. This mapping can also cause display differences in the placement of alphanumeric field data over the graphics display and in the appearance of the filled areas of the graphic display. When the HostRast option is not active, aspect ratio distortion of the displayed graphics appears, unless the aspect ratio of the X Window System display is the same as the IBM 3279.

Color Mixing
GDDMXD/VM supports only the overpaint foreground color mix mode. The initial color of the image area is black, and mixing with the actual background colors is not performed.

An exception is made for data passed by an image data order. In this case, a combined foreground color mix mode is supported, if the series of begin image orders have exactly the same parameter values.

When the HostRast option is active, color mixing is performed by GDDM, not the X Window Display System, so the preceding exception does not apply.

z/VM Considerations

Before you can use GDDM applications with GDDMXD/VM on z/VM, issue the following command to ensure that GETMAIN requests by GDDMXD/VM are processed correctly by CMS.

```
SET STORECLR ENDCMD
```

The SET STORECLR ENDCMD command has no parameters.

GDDMXD Command

Purpose

Use the GDDMXD command to activate GDDMXD/VM so you can send GDDM data to an X Window System display.

Operands

ON Enables GDDMXD/VM. GDDM output is sent to the X Window System display. The system responds with `GDDMXD/VM active`.

OFF
Disables GDDMXD/VM. The system erases the file that was created when GDDMXD/VM was activated and responds with GDDMXD/VM inactive. This is the default.

Usage Notes

- If you do not want to run GDDM applications through the X Window System, do not enable GDDMXD/VM.

Configuring the GDDMXD/VM Environment

You must set several CMS global variables before running GDDM programs through GDDMXD. To create the proper environment, issue the following commands.

```
GLOBAL LOADLIB SCEERUN
GLOBAL TXTLIB GDDMXD ADMNLIB ADMPLIB ADMGLIB ADMHLIB X11LIB COMMTXT SCEELKED
```

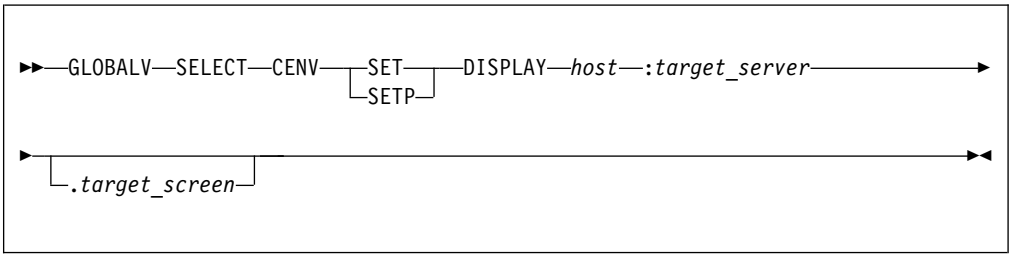
Note: If GDDM/VM Version 2 is in use, omit ADMHLIB from the above GLOBAL TXTLIB command.

These statements force GDDM applications to recognize GDDMXD's presence and allow GDDM output to be directed to the specified X Window System.

Identifying the Target Display

The X Window System uses the C environment variable **DISPLAY** to identify the internet address of the target display.

GLOBALV Command



Purpose

Use the The CMS GLOBALV command to set the required environment variable.

Operands

SELECT CENV
The GLOBALV group that contains C environment variables.

SET
Defines the variable for the current IPL only.

SETP
Defines the variable permanently.

DISPLAY
The C environment variable to be defined.

host
The name or IP address of the host machine running the X Window System server.

target_server
Specifies the number of the display server on the host machine.

target_screen
Specifies the screen to be used on the same *target_server*.

Examples

The following is an example of a Geometry option.

```
GLOBALV SELECT CENV SET DISPLAY charm.cambridg.ibm.com:0.0
```

or

```
GLOBALV SELECT CENV SET DISPLAY 129.42.3.109:0.0
```

Use the workstation xhost command to allow GDDMXD access to the workstation you have selected. To send GDDM output to your selected X Server, enter the following:

```
xhost vm_host_name
```

User-Specified Commands

The user-specified options for GDDMXD/VM are entries in a file called X DEFAULTS. The X DEFAULTS file is searched during initialization of GDDMXD/VM.

Note: The values in the X DEFAULTS file are case sensitive and must be entered as shown.

The following GDDM commands are supported by GDDMXD/VM:
- Geometry
- GColornn
- GMCPnn
- ZWL
- XSync
- CMap
- HostRast
- XClConn
- Compr
- PDTrace
- ANFont*n*
- Enter
- NewLine

Resizing the GDDMXD Graphics Window

GDDMXD supports four graphics window sizes. The initial window size is determined by the window width specified in the Geometry option of the X DEFAULTS file. Subsequently, the window size used by GDDMXD is determined by the width of the window resulting from any resizing that you may have done. The relationship of graphics window size (GDDMXD Graphics Presentation Space) to the actual window width is shown in Table 35 on page 361.

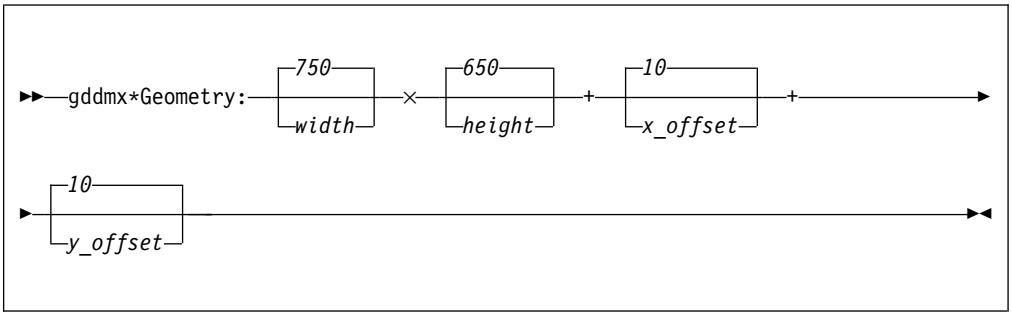
Table 35. Window Width and Window Size

Window Width (pixels)	GDDMXD Graphics Presentation Space (pixels)
< 650	480 horizontal by 352 vertical
>=650 to < 850	720 horizontal by 512 vertical
>= 850 to <= 1024	960 horizontal by 682 vertical
> 1024	1200 horizontal by 864 vertical

For sizes other than the default graphics presentation space size of 720 pixels by 512 pixels, bitmapped data such as symbol sets and images are expanded or contracted to meet the scaling requirements of the specified window size.

Expansion is accomplished by duplicating rows and columns of the data and can result in a slightly different appearance than when viewed at the default size. Contraction is implemented for single-plane data by combining certain rows and columns of the data with a logical OR function. Because this may not yield acceptable results when a black on white image is viewed, a user option, Compr, is provided to specify that a logical AND function be used to contract the data. Contraction of multiplane bitmapped data is accomplished by eliminating certain rows and columns. Data compression results in a different visual appearance than when viewed at the default size. For more information about using this option, see "gddmx*Compr: Command" on page 366.

gddmx*Geometry: Command



Purpose

Use the gddmx*Geometry command to specify the size and location of the initial GDDM graphics window.

Operands

width
Specifies the width of the X-Window in pixels. The default width is 750 pixels.

height
Specifies the height of the X-Window in pixels. The default height is 650 pixels.

x_offset
Specifies the location of the upper left corner of the window where *x_offset* is the horizontal offset from the upper left corner of the display. The default horizontal offset is +10 pixels.

y_offset
Specifies the location of the upper left corner of the window where *y_offset* is the vertical offset from the upper left corner of the display. The default vertical offset is +10 pixels.

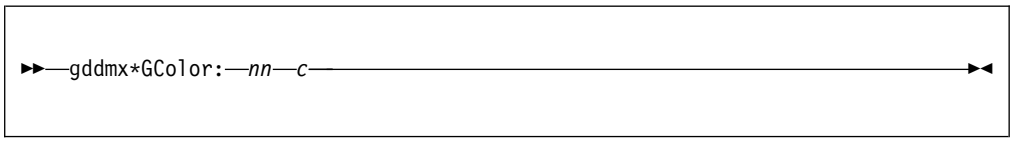
Examples

The following is an example of a Geometry option.

```
gddmx*Geometry: 600x400+20+20
```

In the above example, the Geometry entry in the X DEFAULTS file specifies the size of the initial GDDM graphics window.

gddmx*GColornn: Command



Purpose

Use the gddmx*GColornn command to provide a default mapping of GDDM colors (GColors) to X Window System colors.

Operands

nn Specifies the GDDM color entry that is mapped.

c Specifies the X Window System color that is used as the GDDM color.

Examples

The following is an example of a GColornn option.

```
gddmx*GColor3: Pink
```

In this example, specifying the `GColor3` entry in the X DEFAULTS file maps the GDDM color, Magenta, to the X Window System color, Pink.

GDDMXD/VM provides a default mapping of GDDM colors (GColors) to X Window System colors. If you want to override a default color name or if a default color name is not available by your X Window System server, add a GColornn entry in the X DEFAULTS file. Table 36 lists the GDDM colors that GDDMXD/VM maps to the X Window System.

Table 36. GColors

GColornn	GDDM Color	X Window System Color
GColor1	Blue	Blue
GColor2	Red	Red
GColor3	Magenta	Magenta
GColor4	Green	Green
GColor5	Turquoise	Turquoise
GColor6	Yellow	Yellow

Table 36. GColors (continued)

GColornn	GDDM Color	X Window System Color
GColor7	White	White
GColor8	Black	Black
GColor9	Dark Blue	Dark Slate Blue
GColor10	Orange	Orange
GColor11	Purple	Plum
GColor12	Dark Green	Dark Green
GColor13	Dark Turquoise	Dark Turquoise
GColor14	Mustard	Wheat
GColor15	Gray	Gray
GColor16	Brown	Brown

gddmx*GMCPnn: Command

Purpose

Use the gddmx*GMCPnn command to override GDDM multicolor patterns (GMCP) with workstation color names.

Operands

nn Specifies the GDDM multicolor pattern.

c Specifies the X Window System color that is used as the GDDM color.

Examples

The following is an example of a GMCPnn option.

```
gddmx*GMCP126:  MediumBlue
```

In the preceding example, the color `MediumBlue` is used when multicolor pattern 126 is specified by the GDDM application.

gddmx*ZWL: Command

Purpose

Use the gddmx*ZWL command to specify the line widths of the X Window System.

The X Window System supports a range of line widths. Because some X Window System servers draw wide lines at a slow rate, you can use the Zero Width Lines (ZWL) option to increase the speed at which GDDMXD/VM draws lines.

ZWL directs GDDMXD/VM to draw all lines using zero width lines. The X Window System server uses the fastest drawing algorithm to draw the lines even though the resulting line may not be exactly the same as if it had been drawn as a wide line.

Operands

Y Directs GDDMXD/VM to use zero width lines for all drawing.

N Directs GDDMXD/VM to use normal width lines for all drawing. This is the default.

gddmx*XSync: Command

Purpose

Use the gddmx*XSync command to requests that the X Window System process one request at a time.

The X Window System operates asynchronously. By the time an error has been detected, the application may have issued more requests.

Operands

Y Directs GDDMXD/VM to request the X Window System to operate synchronously.

N Allows the X Window System to operate asynchronously. This is the default.

gddmx*CMap: Command

Purpose

Use the gddmx*CMap command to specify whether or not the XInstallColormap is loaded.

During initialization, GDDMXD/VM issues the X Window System call, XInstallColormap, to load the default color map (CMap). If the CMap option is specified as N, the XInstallColormap call is not made. This option accommodates the X Window System servers that load their own color maps and do not want the clients to load different color maps.

Operands

Y Directs GDDMXD/VM to load the default colormap. This is the default.

N Directs GDDMXD/VM to bypass loading the default colormap.

gddmx*HostRast: Command

Purpose

Use the gddmx*HostRast command to enable you to perform the raster image processing at the VM host.

The default device model for GDDMXD/VM is an IBM 3179G Color Graphic Display Station with a mouse and raster image processing is performed at the workstation.

When the HostRast option `Y` is specified, GDDM performs the raster image processing and transmits the picture as a series of characters whose pixel definitions have been transmitted to Programmed Symbol Sets. The picture is mapped exactly as an IBM 3279 Color Display Station.

Operands

Y Directs GDDMXD/VM to use the IBM 3279 as a device model and perform raster image processing on the VM host.

N Directs GDDMXD/VM to use the IBM 3179G as a device model and perform raster image processing on the workstation. This is the default.

Usage Notes

- The APL2® character set is not supported when the HostRast option is active.

gddmx*XClConn: Command

Purpose

Use the gddmx*XClConn command to instruct GDDMXD/VM not to close the connection to the X Window System when the GDDM device is closed.

The default operation of GDDMXD/VM creates the GDDMXD/VM window when the GDDM application opens the display device, destroys the window, and breaks the connection to the X Window System display when the application closes the display device.

Some applications can repeatedly close and reopen the display device, which can cause a delay of several seconds to reestablish the connection to the X Window System display. It is no longer necessary to use this option since GDDM will only close the X Window environment after a GDDM FSTERM call has been issued. All GDDM applications should issue the FSTERM call to close the environment. The option remains for the sake of compatibility with existing and older applications.

Operands

Y Instructs GDDMXD to close the connection to the X Window System when the display device is closed by the GDDM application. This is the default.

N Instructs GDDMXD to leave the connection to the X Window System active when the display device is closed by the GDDM application. With this parameter, GDDMXD cannot destroy the window and close the connection to the workstation when the application is finished with the device. The last GDDM graphics window created by the application is displayed until a GDDM application is started on the same VM host for the same workstation, or until you explicitly destroy the window using the workstation window manager facilities.

gddmx*Compr: Command

Purpose

Use the gddmx*Compr command to control the technique used to compress (Compr) bitmapped data when a graphices window size of 480 by 352 pixels is in use.

Operands

Or 'O' or 'o' specifies that an OR function should be used. This is the default value.

And
'A' or 'a' specifies that an AND function should be used when compressing bitmapped data.

gddmx*PDTrace: Command

Purpose

Use the gddmx*PDTrace command in your X DEFAULTS file is used to generate a GDDM problem determination trace (PDTrace).

Operands

Y Generates GDDMXD trace information.

N Does not generate the GDDMXD trace information. This is the default.

gddmx*ANFontn Command

Purpose

Use the gddmx*ANFont command to specify the X Window System font that GDDMXD should use for displaying characters in the alphanumeric presentation space of the GDDMXD window.

Graphics mode 1 and 2 characters in the graphics presentation space are not affected by this option. The value of *n* ranges from 1 to 4 and defines the X Window System font for each of the four sizes of presentation space supported by GDDMXD. The ANFont*n* option can be specified for any, all, or none of the four values of *n*.

Operands

n Specifies the size of the presentation space.

fontname
Specifies the name of the X Window System font.

Examples

The X Window System fonts specified should be fixed-spaced fonts whose characters fit into the character box size required by each of the four presentation space sizes shown in Table 37.

Table 37. X Window System Fonts

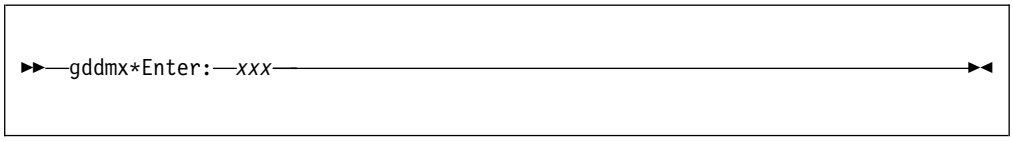
n	Presentation Space	Character Box	Example Font
1	480 x 352	6 x 11	Rom8
2	720 x 512	9 x 16	Rom11
3	960 x 682	12 x 21	Rom14
4	1200 x 864	15 x 27	Rom17

Selecting a font whose characters are larger than the character box size can result in characters overlapping when displayed.

Remapping the Enter and Newline Keys

GDDMXD/VM provides the following commands to allow remapping of the "Enter" and "Newline" keys by the GDDMXD/VM user.

gddmx*Enter: Command



Purpose

Use the gddmx*Enter command in the X DEFAULTS file to identify which X Window System Keysym is to be mapped to the Enter function

The use of this option overrides the default mapping of the Keysym XK_Execute to the Enter function.

Operands

xxx
Specifies the X Window System Keysym representing the physical key. For standard Keysyms, the `XK_` prefix is not included in specifying the option.

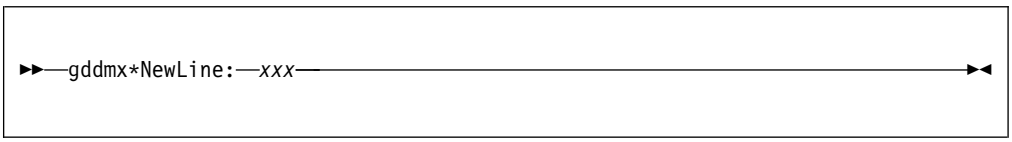
Examples

The following is an example of the Enter Option:

```
gddmx*Enter: Return
```

In the preceding example, the X Window System Keysym, XK_Return is mapped to the Enter function.

gddmx*NewLine: Command



Purpose

Use the gddmx*NewLine command in the X DEFAULTS file to identify which X Window System Keysym is to be mapped to the NewLine function

The use of this option overrides the default mapping of the Keysym XK_Return to the Newline function.

Operands

xxx
Specifies the X Window System Keysym representing the physical key. For standard Keysyms, the `XK_` prefix is not included in specifying the option.

Examples

The following is an example of the NewLine command:

```
gddmx*NewLine: KP_Enter
```

In the preceding example, the X Window System Keysym, KP_Enter is mapped to the NewLine function.

GDDMXD/VM Keyboard Functions

When you enter input to GDDM by GDDMXD/VM, use the following keyboard functions.

- All alphanumeric keys
- F1—F24
  If F13—F24 are not available, use Shift + F1 to Shift + F12
- Tab or Shift + Tab
- Directional arrows
- End key to erase to the end of the field
- Insert key and Delete key
- PA1, PA2, and PA3
- Enter key
- Newline key

Note: The Backspace key is treated as a cursor left key.

If you cannot locate these keys on your workstation, see your workstation X Window System documentation to determine the mapping of the X Window System key symbol definitions to the physical keys.

GDDMXD/VM to X Window System Keyboard Functions

The following are the GDDMXD/VM keyboard functions that translate to X Window System key symbol definitions. Key functions not listed are not supported.

GDDMXD/VM Keyboard Function
X Window System Key Symbol

F1—F12
XK_F1—XK_F12

Tab XK_Tab

Up XK_Up

Down XK_Down

Left XK_Left

Right XK_Right

End XK_End

Insert XK_Insert

Delete XK_Delete

PA1 XK_Prior

PA2 XK_Next

PA3 XK_Home

Clear XK_Pause

Enter XK_Execute

APL2 char set toggle
XK_Backspace with state Mod1Mask

Newline
XK_Return

APL2 Character Set Keyboard

The APL2 character set is activated by simultaneously pressing the X Window System XK_Backspace key (usually the Backspace key) and the State Mod1Mask key (usually the Alt key). For example, if you use the IBM 101 Enhanced Keyboard, the APL2 character set is toggled on and off by pressing and holding Alt, and then pressing Backspace.

When the APL2 character set is active, the characters `APL` are displayed in the title bar of the GDDMXD/VM window.

In the X Window System, a keycode is assigned to each key on the keyboard. GDDMXD/VM uses keycodes in combination with modifier keys. For example, the Shift and Alt keys determine the data that should be passed back from GDDMXD/VM to the X Window System application to identify the user's keystroke data.

A default mapping for the APL2 character set is provided in GDDMXD/VM, which corresponds to the IBM 101 Key Enhanced Keyboard. You can override this default mapping by creating a file called GDXAPLCS MAP to define the mapping

for your workstation. When GDDMXD/VM is initialized, the system searches for a file called GDXAPLCS MAP. If the GDXAPLCS MAP file exists, the data in the GDXAPLCS MAP file replaces the default mapping for all keys.

For additional information about the mapping values defined in the GDXAPLCS MAP file, see Appendix C, "Mapping Values for the APL2 Character Set," on page 455.

Setting Up GDXAPLCS MAP

The following steps describe how to set up the GDXAPLCS MAP file.

1. Use the sample program, KEYCODE module, to determine the keycodes for the physical keyboard keys.

   When the KEYCODE program is executed from your workstation session to the host system, the keycode is displayed for each key depressed at the workstation. Therefore, you can establish the association between a physical key and the character you want to generate.

   For more information about the mapping values that are defined in the GDXAPLCS MAP file, see Appendix C, "Mapping Values for the APL2 Character Set," on page 455.
2. Copy GDXAPLCS SAMPMAP to GDXAPLCS MAP.
3. Edit GDXAPLCS MAP to establish the association between the keycodes in KEYCODE and the character set and code values in Appendix C, "Mapping Values for the APL2 Character Set," on page 455.

Collecting Problem Determination Information

If you are reporting a problem to the IBM Service Representative, you should provide the following information:

- Environment description
- GDDM trace
- GDDMXD problem determination trace
- Spooled console output

Note: The GDDM and GDDMXD problem determination traces must be in machine-readable form. To minimize the size of these traces, find the shortest sequence of actions that reproduce the problem before obtaining traces. Running with these traces activated reduces system performance.

GDDMXD/VM—X Window System Server Environment

You should include the following information in the environment description:

- Host system software name and level
- X Window System server workstation hardware and software names and levels

Obtaining the GDDM Trace

To generate the GDDM trace on the virtual printer, the following lines should be included in the PROFILE ADMDEFS file.

```
*ADMMDFT TRCESTR='FLOW'
*ADMMDFT TRCESTR='PARMSF'
*ADMMDFT TRCESTR='FULLTCA'
*ADMMDFT CMSTRCE=(,)
```

To activate the trace, you should change the asterisk (*) in column 1 of each line to a blank. You should include the FULLTCA line to provide all the information about GDDMXD/VM. You should provide the GDDM trace in machine readable format to your IBM Service Representative.

Obtaining the GDDMXD Problem Determination Trace

To activate the GDDMXD problem determination trace, add the following entry to your X DEFAULTS file.

```
gddmx*PDTrace: Y
```

By default, a file called GDDMX TRACE is written on your A disk. By issuing a CMS FILEDEF command for the data definition name (ddname) GDXDPDT, you can direct the GDDMX TRACE to another device, including the virtual printer.

If the problem you are experiencing results in an X Window System error message, activate the GDDMXD XSync option by including the following entry in your X DEFAULTS file.

```
gddmx*XSync: Y
```

GDDMXD/VM Problem Determination Examples

This section describes the process used to determine the cause of the problems that can occur when you run a GDDM application with GDDMXD/VM.

Problem: Messages from the X Window System or from GDDMXD/VM indicating the connection could not be established.

Explanation: There are problems establishing the connection to the X Window Server.

User response:

- If you received an X Window System message indicating that the server was not authorized to connect to the host, issue an XHOST command at the server specifying the desired host. If the problem persists, seek assistance to verify that the host system and the work station have been set up correctly.
- Is the global variable defining the target X Window System server set up correctly? Issue the following command: GLOBALV SELECT CENV LIST DISPLAY. Is the response correct? If there is more than one display at the server, is the global variable pointing to the correct display? Try using the PING command to check the path to the server. Try an X Windows System application that does not use GDDMXD/VM, for example, XCALC. Try running the GXDEMOn sample GDDMXD programs. Correct any difficulties before trying other GDDM applications.

Problem: Any other problem such as program check or ABEND.

Explanation: There are problems establishing the connection to the X Window Server.

User response: Collect problem determination information.

Problem: An X Window System message indicating an error or any unexpected ABEND.

Explanation: The connection is established, but the GDDMXD window is not displayed.

User response:

- Try running with a different server.
- Try running without a window manager at the server.
- If the X Window System error message is: **XIO: fatal IO error nn (broken pipe)** and the message persists, the problem is most likely in the X Window System and not in GDDMXD/VM. If the error persists, collect problem determination information.

Problem: GDDMXD window is created, but picture drawn is incorrect or user interaction with application produces unexpected results.

User response:

- Is the server at the correct level?
- Is the ZWL option being used? If so, try turning it off.
- Is a supported version of GDDM being used?
- Is the application using GDDM functions that are not supported?
- If the error persists, collect problem determination information.

Demonstration Programs

The demonstration programs display examples of the GDDM functions. Each program displays a series of frames. To move to the next frame, press **Enter**.

GXDEMO1

The following describes the GXDEMO1 frames.

Frame	Description
1	GDDM line types and widths
2	GDDM System Markers
3	Simple picture
4	Circular Arcs
5	Elliptic Arcs
6	2-Part Polyfillet
7	5-Part Polyfillet

GXDEMO2

The following describes the GXDEMO2 frames.

Frame	Description
1	Solid area fill
2	Overlapped polygon fill
3	2-Part graphic area fill
4	GDDM System Shading Patterns
5	Output from GSIMG statement
6	Output from GSIMGS statement

GXDEMO3

The following describes the GXDEMO3 frames.

Frame	Description
1	Three GDDM Text Modes
2	Proportional Spacing
3	GDDM Character Angle
4	GDDM Character Direction
5	GDDM Character Shear

GXDEMO4

The GXDEMO4 frames display the 22 standard GDDM fonts.

GXDEMO4A

The GXDEMO4A frames display a subset of the standard GDDM fonts.

GXDEMO5

The following describes the GXDEMO5 frames.

Frame	Description
1	The Four Segment Transformations
2	Example transformation (Part 1)
3	Example transformation (Part 2)
4	Examples of Called Segments

GXDEMO6

The following describes the GXDEMO6 frames.

Frame	Description
1	PGF Line Chart
2	PGF Surface Chart
3	PGF Histogram
4	PGF Bar Chart
5	PGF Tower Chart
6	PGF Polar Chart
7	PGF Composite Bar Versus Multiple Pie Charts
8	PGF Pie Charts
9	PGF Venn Diagram

Chapter 13. Managing TCP/IP Network Resources with SNMP

Simple Network Management Protocol (SNMP) is used to monitor your TCP/IP network resources.

SNMP defines an architecture that consists of network management stations (SNMP clients), network elements (hosts and gateways), and network management agents and subagents, which perform the information management functions.

SNMP allows clients and agents to communicate network management information through network elements, which act as servers.

An SNMP client is a network workstation that executes management applications that are used to monitor and control network elements. When you use SNMP with TCP/IP, you require NetView® to provide end-user interface to the SNMP client. A NetView operator can use the SNMP command to communicate with SNMP agents. NetView acts as an SNMP client.

Note:

1. VM SNMP supports Management Information Base (MIB) variables and MIB-II variables. For more information about MIB-II variables, see RFC 1158 and Appendix E, "Management Information Base Objects," on page 467.
2. The VM SNMP agent does not support the SET operation. If you use the SET command with the variables listed in Appendix E, "Management Information Base Objects," on page 467 for a VM Agent, you receive a read-only error.

Sample Command Lists

Two sets of sample NetView command lists with a filetype of NCCFLST are supplied on the Samples tape. One set is written in CLIST (SNMPRUN), and the other set is written in REXX (SNMPMGMT).

You should use CLISTs if your host system does not support REXX. The REXX programs supply the same functionality as the CLISTs. These command lists enable you, through revision and modification, to develop a set of NetView/SNMP client/user interface pannels that are customized to your needs.

You can issue SNMP requests with the two sets of sample command lists that are shipped with the TCP/IP product, with command lists that you write, or directly from the command line. The SNMPRUN and the SNMPMGMT commands invoke the main panel from which a NetView operator can execute most of the SNMP requests in full-screen mode. The SNMPRUN command uses a set of command lists written in NetView Command List language; the SNMPMGMT command uses a set of REXX command lists.

For more information about these sample command lists, see the SNMPCLST README and SNMPREXX README files on the client common disk (TCPMAINT 592).

© Copyright IBM Corp. 1987, 2016

SNMP/NetView Overview

Figure 60 illustrates how the interface between the NetView and TCP/IP virtual machines is set up. An additional virtual machine running a special server called the Query Engine(SNMPQE) actually implements the communication function.

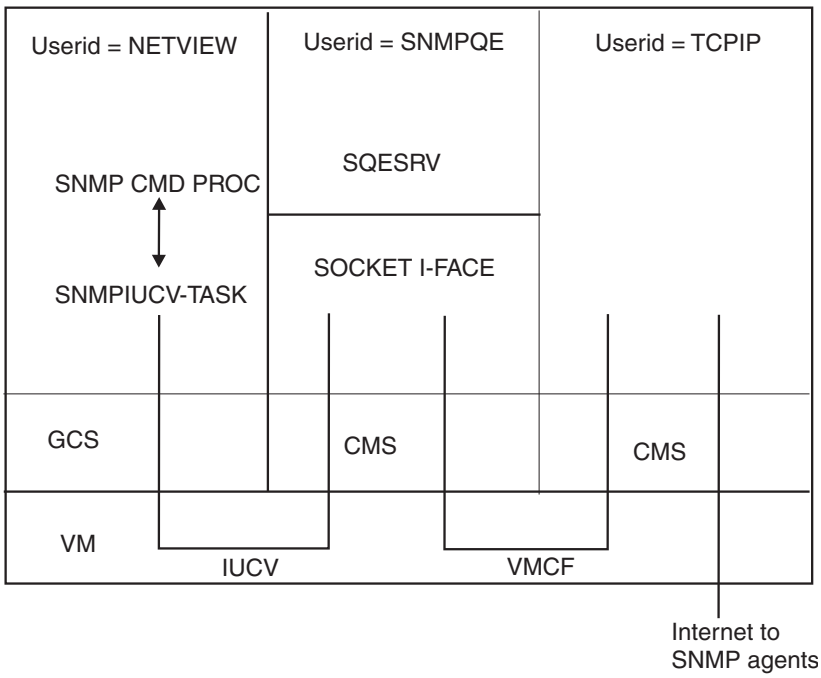


Figure 60. Overview of NetView SNMP Support

The Query Engine communicates with an optional subtask, called SNMPIUCV, running under NetView. It also communicates with TCP/IP. This is done through socket interfaces that are based on the IUCV cross-memory facility.

The SNMP Query Engine uses three types of sockets:

- Datagram sockets through which the SNMP requests and responses flow and traps are received, because the management protocol is based on UDP
- A raw socket into ICMP for the implementation of the PING facility, which is an ICMP echo application
- A stream socket for communication with NetView

To use the interface, a NetView operator or command list issues the SNMP command with the appropriate parameters. This invokes the SNMP command processor, which validates the syntax of the input and, if no errors are found, it queues the request to the SNMPIUCV task. Then, the task passes the request to the SNMP Query Engine, which validates the request, encodes it in ASN.1 format, and builds the protocol data unit (PDU) that is sent to the SNMP agent.

When the Query Engine receives a response from the agent, it decodes the PDU and passes it to the SNMPIUCV NetView task. The task converts the response into a multiline message. The messages are prefixed with SNM, so that they can be assigned to specific NetView operators or autotasks.

SNMP Commands

To issue an SNMP request, use the SNMP command. The following is a list of the SNMP commands you can use on NetView.

Note: The SNMP commands can be abbreviated; the shortest acceptable form appears in uppercase.

The SNMP Query Engine issues SNMP requests to the agents and processes SNMP responses returned by the agents. The agents can also forward unsolicited messages, known as traps, to NetView and other clients.

NetView can use the following SNMP commands.
- SNMP Get
- SNMP GETNext
- SNMP Set
- SNMP TRAPson
- SNMP TRAPSOFf
- SNMP MIBvname
- SNMP PING

The following sections describe these commands.

SNMP Commands Overview

The following provides information about SNMP commands.

- When the SNMP command is issued from the NetView Command Facility command line, all input is translated to uppercase (standard NetView) before it is sent to the SNMP Query Engine.
- When the SNMP command is issued from a CLIST, input is passed in whatever case it was passed from the CLIST (for example, mixed case).
- The short names for the variables passed to the query engine are compared against the entries in the MIB_DESC DATA file in a case insensitive way. For more information about this file, see *z/VM: TCP/IP Planning and Customization*.
- The community name is passed to the SNMP agent in the same case as it was received by the query engine.
- If multiple variables are specified with the GET, GETNEXT, or SET commands, they are all packaged in one SNMP PDU to be sent to the agent.
- For PING, you can specify only one *host_name*.
- The responses may not be received in the same order they are issued.
- The SNMP agent can receive SNMP requests over any interface.

Return Codes

The following is a list of the return codes generated by SNMP.

Code	Description
1	Error from DSIGET, cannot continue
2	Invalid function specified
3	Missing SNMP function
4	Not enough parameters
5	Missing variable name

6 Missing variable value

7 Missing or invalid host name

8 Missing community name

9 SNMPIUCV not active

10 Error from DSIMQS

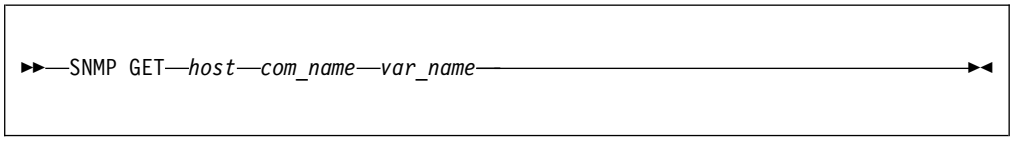
11 Invalid *net_mask*/desired network

12 Missing/Invalid trap *filter_id*

1001+ All return codes above 1001 indicate that the command was successful

The return code represents the sequence number (or *filter_id*) passed to SNMP Query engine. The asynchronous response is identified by this sequence number. For a TRAPSON request, this is the *filter_id* to be used for a subsequent TRAPSOFF request.

SNMP GET Command



Purpose

Use the SNMP GET command to obtain one or more variables from an SNMP agent.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

host
Specifies the name of the destination host where the SNMP request is sent. The host can be specified with its name or with its IP address in dotted-decimal notation.

Note: The SNMP Query Engine treats numbers with leading zeros as octal numbers. Therefore, do not use leading zeros.

com_name
Specifies the community name to use when querying the SNMP agent on the destination host.

Note: This parameter is case sensitive.

var_name
Specifies one or more MIB variable names to be obtained. You can specify the names in short form or in ASN.1 notation (for example, sysDescr.0 or 1.3.6.1.2.1.1.1.0). Currently, a maximum of 10 variables for each request is implemented in the SNMP Query Engine.

Note: This parameter is not case sensitive.

Usage Notes

1. You must specify the host name or IP address of the host where the agent is running, as well as the community name to be used when making the query.
2. You must specify the variable name of each variable to be obtained. All standard MIB and MIB-II variable names (defined in RFC 1156 and RFC 1158) are supported by the VM SNMP client. In addition, you can specify enterprise-specific variables that are defined by the implementer of a particular SNMP agent. See *z/VM: TCP/IP Planning and Customization* for information about how to add the enterprise-specific variables to the MIB_DESC DATA file.
3. All lines do not need to be present, but the first line is always message SNM040I, and the last line is always message SNM049I.
4. If an error was detected, messages SNM042–SNM044 may not be present. You can get (in addition to other messages) error messages in the following form (all as part of multiline message SNM040I).

```
SNM045I Major error code: n
SNM046I Minor error code: y
SNM047I Error index: z
SNM048I Error text: message text
```

   See "Major and Minor Error Codes and SNMP Value Types" on page 393 for information about value types and minor and major error codes.
5. If you issue a GET for multiple variables, messages SNM042 through SNM044 are displayed for each variable.
6. If a variable value is too long, message SNM044 may not fit on an 80-character line. If this happens, the value is split and multiple SNM044 messages are displayed.
7. The SNMP response always displays the variable name in ASN.1 notation. You can use SNMP MIBVNAME to obtain the short name for the variable.For more information, see "SNMP MIBVNAME Command" on page 387.
8. According to RFC 1157, a message exchanged between SNMP entities (including version identification and community name) can be as small as 484 octets. If you specify up to 10 variables in a GET/GETNEXT command, the names may be short enough to send the GET command to the SNMP agent, but the response may be too long to fit in the message. As a result, you receive a `tooBig` error.
9. When you issue a GET for multiple variables, they are returned in the same sequence as requested. In the "Examples," GET was issued for `sysDescr.0 sysObjectID.0 sysUpTime.0.`. The same three variables are returned in the response. If one (or more) of the variables requested results in an error, all variables listed after the first variable in error are ignored, and data is not returned for them.
10. For more information about value types, minor, and major error codes, see "Major and Minor Error Codes and SNMP Value Types" on page 393.
11. For a description of all variables and the meaning of their values, see RFC 1156 and RFC 1158.

Examples

If you know:

```
hostname            -  anyhost
IP address          -  129.34.222.72
community name      -  public
variable name       -  sysDescr.0
asn.1 variable name -  1.3.6.1.2.1.1.1.0
variable name       -  sysObjectID.0
asn.1 variable name -  1.3.6.1.2.1.1.2.0
variable name       -  sysUpTime.0
asn.1 variable name -  1.3.6.1.2.1.1.3.0
```

You can issue the following SNMP GET commands:

```
snmp get 129.34.222.72 public 1.3.6.1.2.1.1.1.0
snmp get 129.34.222.72 public sysDescr.0
snmp get anyhost public 1.3.6.1.2.1.1.1.0
snmp get anyhost public sysDescr.0
snmp get anyhost public sysObjectID.0
snmp get anyhost public sysUpTime.0
snmp get anyhost public sysDescr.0 sysObjectID.0 sysUpTime.0
```

After the SNMP command is completed, you get a message similiar to the following:

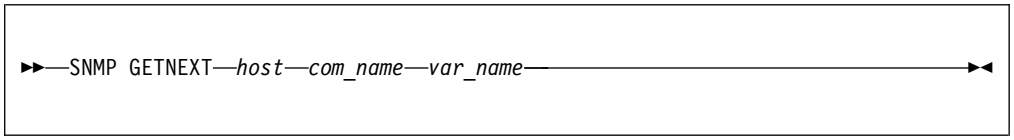
```
SNM050I SNMP Request 1001 from NETOP accepted, sent to Query Engine
```

When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form.

```
SNM040I SNMP Request 1001 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.1.1.0
SNM043I Variable value type: 9
SNM044I Variable value: AIX 2.2.1 SNMP Agent Version 1.0
SNM042I Variable name: 1.3.6.1.2.1.1.2.0
SNM043I Variable value type: 3
SNM044I Variable value: 1.3.6.1.4.1.2.1.1
SNM042I Variable name: 1.3.6.1.2.1.1.3.0
SNM043I Variable value type: 8
SNM044I Variable value: 98800
SNM049I SNMP Request 1001 end of response
```

For a list of variables supported by the VM Agent, see Appendix E, “Management Information Base Objects,” on page 467.

SNMP GETNEXT Command



Purpose

Use the SNMP GETNEXT command in the following format to obtain the next variable in the MIB tree from an SNMP agent.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

host
Specifies the name of the destination host where the SNMP request is sent. The host can be specified with its name or with its IP address in dotted-decimal notation.

Note: The SNMP Query Engine treats numbers with leading zeros as octal numbers. Therefore, do not use leading zeros.

com_name
Specifies the community name to use when querying the SNMP agent on the destination host.

Note: This parameter is case sensitive.

var_name
Specifies one or more MIB variable names to be obtained. You can specify the names in short form or in ASN.1 notation (for example, sysDescr.0 or 1.3.6.1.2.1.1.1.0). Currently, a maximum of 10 variables for each request is implemented in the SNMP Query Engine.

Note: This parameter is not case sensitive.

Usage Notes

1. You must specify the host name or IP address of the host where the agent is running, as well as the community name to be used when making the query. You must specify the name of the variable preceding the desired variable. In addition to the standard MIBs, there may be enterprise-specific variables as defined by the implementer of a particular SNMP agent.
2. The GETNEXT command is used to interrogate a table (for example, the interface table) or an array. You can issue a GETNEXT command at the start of a table (use instance 0.0). The first element in the table is returned. The process continues in a loop, performing GETNEXT requests on the previously obtained variable name, until the name of the variable returned no longer has the same prefix as the one at the start of the table. This condition occurs when the GETNEXT request returns a variable that is in the next group.

Examples

- If you know:

```
hostname            -  anyhost
IP address          -  129.34.222.72
community name      -  public
variable name       -  ifAdminStatus (in ifTable)
asn.1 variable name -  1.3.6.1.2.1.2.2.1.7
```

  You can issue an SNMP GETNEXT command in one of the following ways:

```
snmp getnext 129.34.222.72 public 1.3.6.1.2.1.2.2.1.7.0
snmp getnext 129.34.222.72 public ifAdminStatus.0
snmp getnext anyhost public 1.3.6.1.2.1.2.2.1.7.0
snmp getnext anyhost public ifAdminStatus.0
```

- The GETNEXT command is completed in the same manner as the GET command, and you receive an asynchronous response similar to the following example.

  The first instance of the variable has a status of 1 or greater (ends in 7.1) in this example:

```
SNM040I SNMP Request 1001 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.2.2.1.7.1
SNM043I Variable value type: 1
SNM044I Variable value: 1
SNM049I SNMP Request 1001 end of response
```

You can then issue another GETNEXT command in one of the following ways:

```
snmp getnext 129.34.222.72 public 1.3.6.1.2.1.2.2.1.7.1
snmp getnext 129.34.222.72 public ifAdminStatus.1
snmp getnext anyhost public 1.3.6.1.2.1.2.2.1.7.1
snmp getnext anyhost public ifAdminStatus.1
```

- The GETNEXT command is completed in the same manner as the GET command, and you receive an asynchronous response similar to the following example.

  The second instance of the variable has a status of 1 or greater (ends in 7.2) in this example:

```
SNM040I SNMP Request 1002 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.2.2.1.7.2
SNM043I Variable value type: 1
SNM044I Variable value: 1
SNM049I SNMP Request 1002 end of response
```

  You can then issue another GETNEXT command in one of the following ways:

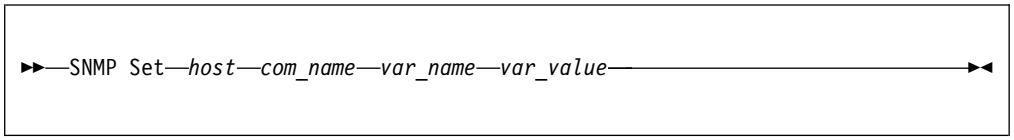
```
snmp getnext 129.34.222.72 public 1.3.6.1.2.1.2.2.1.7.2
snmp getnext 129.34.222.72 public ifAdminStatus.2
snmp getnext anyhost public 1.3.6.1.2.1.2.2.1.7.2
snmp getnext anyhost public ifAdminStatus.2
```

- The GETNEXT command is completed in the same manner as the GET command, and you receive an asynchronous response similar to the following:

```
SNM040I SNMP Request 1003 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.2.2.1.8.1
SNM043I Variable value type: 1
SNM044I Variable value: 1
SNM049I SNMP Request 1003 end of response
```

  The returned variable is the first entry in the next instance, which has a value of 1 or greater (ends in 8.1 rather than 7.x). The returned variable indicates that the end of the table for the ifAdminStatus variable has been reached.

SNMP SET Command



Purpose

Use the SNMP SET command to set or change the value of one or more variables in an SNMP agent.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

host
Specifies the name of the destination host where the SNMP request is sent. The host can be specified with its name or with its IP address in dotted-decimal notation.

Note: The SNMP Query Engine treats numbers with leading zeros as octal numbers. Therefore, do not use leading zeros.

com_name
Specifies the community name to use when querying the SNMP agent on the destination host.

Note: This parameter is case sensitive.

var_name
Specifies one or more MIB variable names to be obtained. You can specify the names in short form or in ASN.1 notation (for example, sysDescr.0 or 1.3.6.1.2.1.1.1.0). Currently, a maximum of 10 variables for each request is implemented in the SNMP Query Engine.

Note: This parameter is not case sensitive.

var_value
Specifies the value(s) to be stored in the variable(s).

Usage Notes

- You must specify the host name or IP address of the host where the agent is running, as well as the community name to be used. The community name used for a SET request is frequently different than the community name for a GET request. You must specify the names and values of each variable to be set. RFC 1156 and RFC 1158 define the variables that you can set with read-write access. In addition, there may be enterprise-specific variables as defined by the implementer of a particular SNMP agent.

Examples

- If you know:

```
hostname            -  anyhost
IP address          -  129.34.222.72
community name      -  publicw
variable name       -  ifAdminStatus
asn.1 variable name -  1.3.6.1.2.1.2.2.1.7.1
     (instance 1)
```

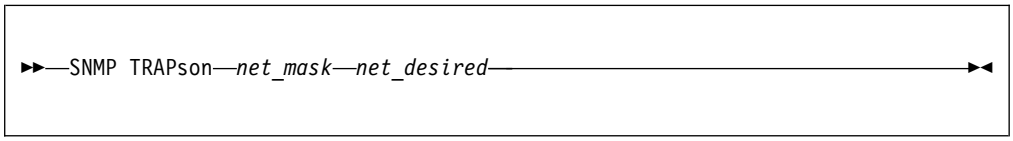
  You can then issue an SNMP SET command in one of the following forms to set the administrative status of the first interface in the ifTable (first instance) to test.

```
snmp set 129.34.222.72 publicw 1.3.6.1.2.1.2.2.1.7.1 3
snmp set 129.34.222.72 publicw IfAdminStatus.1 3
snmp set anyhost publicw 1.3.6.1.2.1.2.2.1.7.1 3
snmp set anyhost publicw ifAdminStatus.1 3
```

- When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form.

```
SNM040I SNMP Request 1001 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.2.7.1
SNM043I Variable value type: 1
SNM044I Variable value: 3
SNM049I SNMP Request 1001 end of response
```

SNMP TRAPSON Command



Purpose

Use the SNMP TRAPSON command to request that the SNMP Query Engine forward SNMP traps to NetView.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

The SNMP Query Engine can forward only those traps that it receives. Each agent has a trap destination table, which lists all the hosts that should receive that agent's traps. The host name of your system should be in the trap destination table of all agents from which you want to receive traps.

Operands

net_mask
: Specifies, in dotted-decimal notation, the network mask to be evaluated with the IP address of incoming traps. The dotted decimal IP address is ANDed with this mask.

net_desired
: Specifies the network from which you want to receive traps. When you request traps using the SNMP TRAPSON command, it returns a request number of *filter_id*, which the SNMP Query Engine associates with the TRAPSON request. To stop receiving traps, specify this *filter_id* in the TRAPSOFF request.

var_name
: Specifies one or more MIB variable names to be obtained. You can specify the names in short form or in ASN.1 notation (for example, sysDescr.0 or 1.3.6.1.2.1.1.1.0). Currently, a maximum of 10 variables for each request is implemented in the SNMP Query Engine.

  Note: This parameter is not case sensitive.

This command permits the specification of a filtering condition, which enables the Query Engine to perform filtering.

The SNMP TRAPSON command assigns a unique request number to each filter (also called a *filter_id*) and returns this number in a message and in the return code. This *filter_id* is the argument to an SNMP TRAPSOFF command, which is used to stop receiving traps that pass this filter.

Usage Notes

1. In the response to the SNMP TRAPSON request, not all lines need to be present; but the first line is always message SNM040I, and the last line is always message SNM049I.

2. For the multiline trap message, not all lines need to be present; but the first line is always message SNM030I, and the last line is always message SNM039I.
3. Additional messages (SNM036I-SNM038I) may be present if the trap has additional data.
4. If a variable value is too long, message SNM038 may not fit on an 80-character line. If this happens, the value is split and multiple SNM038 messages are displayed.
5. The SNMP trap data always displays the variable name in ASN.1 notation. You can use SNMP MIBVNAME to obtain the short name for the variable.
6. A trap always shows the agent address in the form of an IP address in dotted-decimal notation.
7. See "Major and Minor Error Codes and SNMP Value Types" on page 393 for information about value types, minor, and major error codes.
8. See Appendix F, "SNMP Generic TRAP Types," on page 495 for a description of the traps and the meanings of the generic trap types.
9. You can issue multiple TRAPSON requests, either with the same or with a different filter. If a trap passes multiple filters, the trap is sent to NetView multiple times. However, in NetView, the header and trailer lines (messages SNM030I and SNM039I) of the duplicate trap are different, because they contain the *filter_id* (request number) by which the trap was forwarded. Different types of traps from different hosts can have the same *filter_id*, if these traps pass the same trap filter. If an SNMP request is issued with the wrong community name, it receives three AUTHENTICATION FAILURE traps with the same *filter_id* but different time stamps from the same host. This is because the SNMP Query Engine tries to send the same request three times if a response is not received from the host, and each attempt causes the host to generate an AUTHENTICATION FAILURE trap.
10. Once the TRAPSON command has been issued, traps can start to arrive asynchronously. They can even arrive after the operator who issued the TRAPSON command logs off. Often, a TRAPSON command is issued by a CLIST, and the received trap data triggers another CLIST to handle the trap data. Therefore, the messages in the range SNM030 through SNM039 are sent to the *authorized receiver*. For a NetView operator to see the traps, the operator must have the following statement in the profile.

    ```
    AUTH MSGRECVR=YES
    ```

    However, only one operator receives the message. The messages also go to the log file, so you can always browse the log file to see trap data. And as a last resort, you can *assign* trap messages to go to a specific operator using the NetView ASSIGN operator command.

Examples

- If you know:

  ```
  IP address          -  129.34.222.72
  net mask            -  255.255.255.255
  ```

  You can issue the following SNMP TRAPSON commands:

  ```
  snmp trapson
  snmp trapson 255.255.255.255 129.34.222.72
  ```

  The first command receives all traps (the default is a mask of 0 and a desired network of 0). The second command only receives traps from a specific host named anyhost.

  After the command is completed, you receive a message similar to the following:

```
SNM050I SNMP Request 1001 from NETOP accepted, sent to Query Engine
```

- When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form to indicate that the TRAPSON request was accepted.

```
SNM040I SNMP Request 1001 from NETOP Returned the following response:
SNM045I Major error code: 0
SNM046I Minor error code: 0
SNM047I Error index: 0
SNM048I Error text: no error
SNM049I SNMP Request 1001 end of response
```

When traps arrive, NetView displays each trap with a multiline message in the following form. This multiline message is sent to the authorized receiver (AUTH MSGRECVR=YES); it may not show up on the console of the operator who issues the TRAPSON command.

```
SNM030I SNMP request 1001 received following trap:
SNM031I Agent Address: 129.34.222.34
SNM032I Generic trap type: 4
SNM033I Specific trap type: 0
SNM034I Time stamp: 472600
SNM035I Enterprise Object ID: 1.3.6.1.4.1.2.1.1
SNM039I SNMP request 1001 End of trap data
```

SNMP TRAPSOFF Command

Purpose

When you have asked the SNMP Query Engine to forward traps to NetView, it keeps doing so until the IUCV connection breaks or until you issue an SNMP TRAPSOFF command.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

filter_id
Specifies the trap filter ID.

When you request traps using the SNMP TRAPSON command, it returns a request number or *filter_id*, which the SNMP Query Engine associates with the TRAPSON request. To stop receiving traps, specify this *filter_id* in the TRAPSOFF request.

The SNMP TRAPSON command assigns a unique request number to each filter (also called a *filter_id*) and returns it in a message as the return code. This *filter_id* can later be used as the argument to an SNMP TRAPSOFF command if you want to stop receiving traps that pass this filter. Only one *filter_id* for each SNMP TRAPSOFF command can be passed. Extraneous arguments are ignored.

Examples

- If you know the filter_id is 1001, you can issue the following SNMP TRAPSOFF command to tell the SNMP Query Engine to quit sending traps that would pass filter 1001.

  ```
  snmp trapsoff 1001
  ```

  The command completes with a message similiar to the following:

  ```
  SNM050I SNMP Request 1001 from NETOP accepted, sent to Query Engine
  ```

- When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form to indicate that the TRAPSOFF request was accepted.

  ```
  SNM040I SNMP Request 1002 from NETOP Returned the following response:
  SNM045I Major error code: 0
  SNM046I Minor error code: 0
  SNM047I Error index: 0
  SNM048I Error text: no error
  SNM049I SNMP Request 1002 end of response
  ```

SNMP MIBVNAME Command

Purpose

When you get an ASN.1 variable name as part of a trap, use the SNMP MIBVNAME command to find the short name of the variable.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

asn.1_name
Specifies the ASN.1 notation of one MIB variable. You can specify only one variable, so additional arguments are ignored.

Usage Notes

1. All lines do not need to be present, but the first line is always message SNM040I, and the last line is always message SNM049I. If an error occurs, an error message explains what is wrong.
2. If an error is detected, messages SNM042 through SNM044 may not be displayed. You receive error messages (in addition to other messages) in the following form (all as part of multiline message SNM040I).

   ```
   SNM045I Major error code: n
   SNM046I Minor error code: y
   SNM047I Error index: z
   SNM048I Error text: message text
   ```

See "Major and Minor Error Codes and SNMP Value Types" on page 393 for information about value types and minor and major error codes.

3. One ASN.1 variable name only can be passed for each SNMP MIBVNAME command. Additional parameters are ignored.

Examples

- If you have a trap that tells you:

```
SNM030I SNMP request 1001 received following trap:
SNM031I Agent Address: 129.34.222.34
SNM032I Generic trap type: 2
SNM033I Specific trap type: 0
SNM034I Time stamp: 472600
SNM035I Enterprise Object ID: 1.3.6.1.4.1.2.1.1
SNM036I Variable name: 1.3.6.1.2.1.2.2.1.1
SNM037I Variable value type: 1
SNM038I Variable value: 2
SNM039I SNMP request 1001 End of trap data
```

You can issue the following SNMP MIBVNAME command to find the short MIB variable name.

```
snmp mibvname 1.3.6.1.2.1.2.2.1.1
```

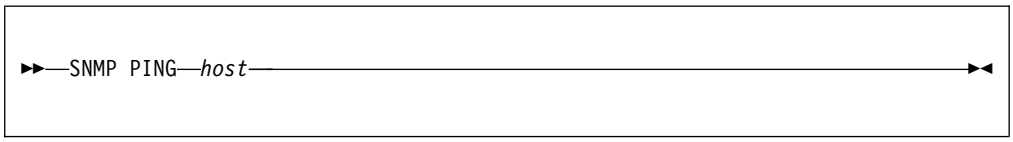
The command completes with a message similar to the following:

```
SNM050I SNMP Request 1002 from NETOP accepted, sent to Query Engine
```

When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form.

```
SNM040I SNMP Request 1002 from NETOP Returned the following response:
SNM042I Variable name: 1.3.6.1.2.1.2.2.1.1
SNM043I Variable value type: 9
SNM044I Variable value: ifIndex
SNM049I SNMP Request 1002 end of response
```

SNMP PING Command



Purpose

Use the SNMP PING command to obtain the minimum round trip response time from the Query Engine to a specific node.

Note: This command is one of the NETVIEW SNMP client commands and NETVIEW is required.

Operands

host
Specifies the name of the destination host where the SNMP request is sent. The host can be specified with its name or with its IP address in dotted-decimal notation.

Note: The SNMP Query Engine treats numbers with leading zeros as octal numbers. Therefore, do not use leading zeros.

The Query Engine issues one PING (an ICMP echo on a raw socket) and returns the value in milliseconds in an IBM-defined SNMP variable minRTT. For more information about the SNMP PING command, see *z/VM: TCP/IP Planning and Customization*. Because only one PING is issued, this is also the average and the maximum response time. If the PING does not respond, the Query Engine retries twice, once after 1 second and again after 2 seconds (Query Engine default retry mechanism). If a response is not received after all retries have been exhausted, a variable value of -1 is returned to indicate that a reply was not received.

Usage Notes

1. All lines do not need to be present, but the first line is always message SNM040I, and the last line is always message SNM049I. If an error occurs, an error message explains what is wrong.
2. If an error is detected, messages SNM042 through SNM044 may not be displayed. You receive error messages (in addition to other messages) in the following form (all as part of multiline message SNM040I).

   ```
   SNM045I Major error code: n
   SNM046I Minor error code: y
   SNM047I Error index: z
   SNM048I Error text: message text
   ```

   See "Major and Minor Error Codes and SNMP Value Types" on page 393 for information about value types and minor and major error codes.
3. One ASN.1 variable name only can be passed for each SNMP MIBVNAME command. Additional parameters are ignored.

Examples

- If you know:

  ```
  nodename            -  anynode
  IP address          -  129.34.222.72
  ```

  You can issue the following SNMP PING commands:

  ```
  SNMP PING ANYNODE
  SNMP PING 129.34.222.72
  ```

  The command completes with a message similar to the following:

  ```
  SNM050I SNMP Request 1001 from NETOP accepted, sent to Query Engine
  ```

  When the response arrives in NetView (asynchronously), NetView displays it as a multiline message in the following form:

  ```
  SNM040I SNMP Request 1001 from NETOP Returned the following response:
  SNM042I Variable name: 1.3.6.1.4.1.2.2.1.3.2.129.34.222.72
  SNM043I Variable value type: 1
  SNM044I Variable value: 26
  SNM049I SNMP Request 1001 end of response
  ```

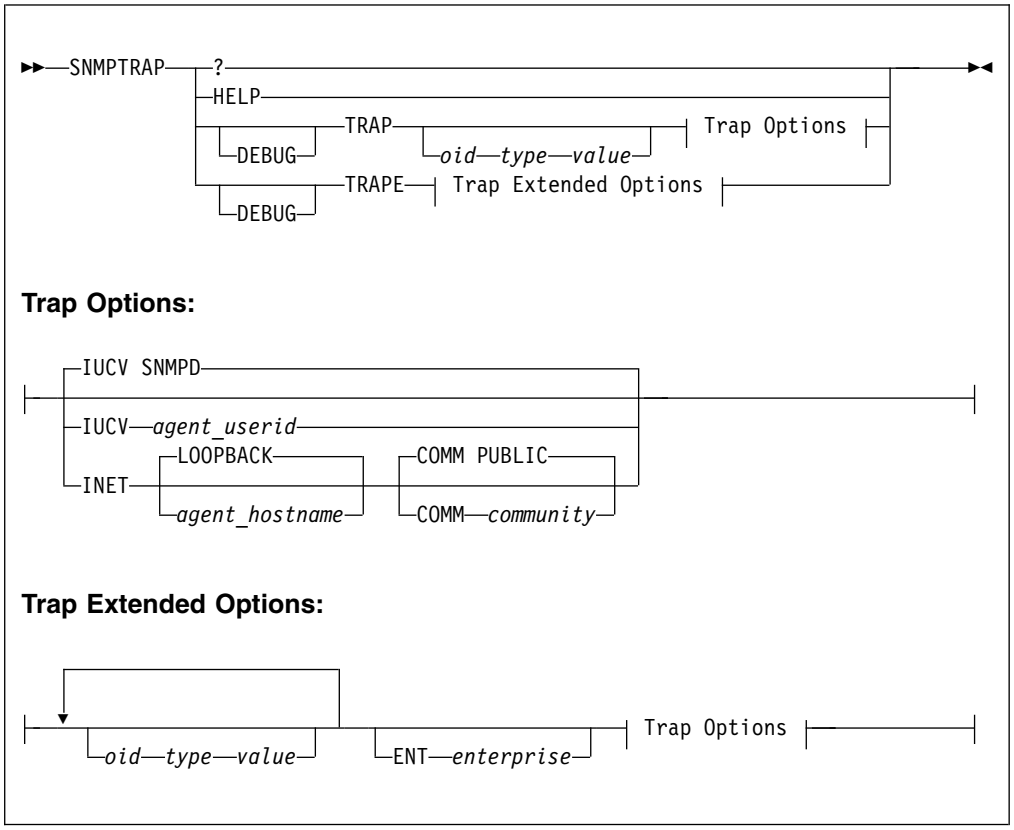
SNMP Native Commands

Traps are unsolicited data messages sent by an SNMP agent to its SNMP managing system. These messages are usually used to inform the managing station about a special condition that has occurred either in an agent system or in the network.

SNMPTRAP is a command line utility for generating SNMP version 1 enterprise-specific traps to report events to the SNMP manager. The SNMPTRAP utility allows you to generate:

- enterprise specific traps
- extended enterprise specific traps

SNMPTRAP Command



Purpose

Use the SNMPTRAP command to generate a notification (trap) to report an event to the SNMP manager.

Operands

HELP or ?
Invokes output with an explanation about how the SNMPTRAP command is used. You cannot place the HELP or ? operand on the SNMPTRAP command line with any other operands.

DEBUG
Specifies that extra messages are to be printed. DEBUG must be the first operand.

TRAP
Generates an enterprise specific trap (generic trap type is 6 for EnterpriseSpecific) with the following options:

oid
The SNMP OID associated with the specific data for the trap generated.

type
The value of *type* can be one of the following (which correspond to the types defined in the include file "snmp_dpi.h")
NUMBER
integer
STRING
hex value of string (each byte is an EBCDIC value)
OID object identifier
EMPTY
any value (ignored)
IPADDR
internet address
COUNTER
non-negative integer
GAUGE
non-negative integer
TICKS
time in 1/100th of seconds since agent initialized
TEXT display string (each byte is an ASCII value)

value
Passes data as an additional value for the variable type specified. For STRING and TEXT data with imbedded blanks, see 1 on page 392. For type EMPTY, a value must be specified on the command line, but it is ignored when the request is created.

TRAPE
Generates an extended enterprise specific trap with the following options:

oid
The SNMP OID associated with the specific data for the trap generated.

type
Same data types as defined for trap above.

value
Passes data values for additional variable bindings. Consists of name-value pairs that further describe the trap notification. For each name-value pair, specify an OID, a value type, and a value. Maximum number of variable bindings is 10.

ENT *enterprise*
Specifies the object identifier, in dot notation, of this trap. This is valid only with TRAPE. If enterprise is not specified, the SNMP agent will place the value of sysObjectID in the enterprise field of the SNMP Trap-PDU.

IUCV
Specifies that an AF_IUCV socket is to be used to connect to the SNMP agent. The IUCV operand is the default.

agent_userid
Specifies the user ID where the SNMP agent (SNMPD) is running. The default is SNMPD.

INET
Specifies that an AF_INET socket is to be used to connect to the SNMP agent.

agent_hostname
Hostname or internet address of the SNMP agent generating the trap.

The default is the local host, or loopback address (127.0.0.1).

COMM *community*
Specifies the community name to get the dpiPort. The default is PUBLIC.

The COMM community option is only valid with INET. The query_DPI_port function is used to obtain the port number on which the DPI capable SNMP agent at the specified host is listening for connections (TCP).

Usage Notes

1. Use double quotation marks (") to enclose STRING and TEXT data which contain imbedded blanks. If STRING and TEXT data with imbedded blanks is entered from the CMS command line and your virtual machine's TERMINAL ESCAPE symbol is set to ", you will need to enclose the string data in two double quotation marks ("") or change your virtual machine's logical escape symbol using the CP TERMINAL ESCAPE command.
2. The target network manager host(s) to which the trap message will be sent is determined by the contents of the SNMPTRAP DEST file. See "Configuring the SNMP Servers" in *z/VM: TCP/IP Planning and Customization*

Return Code

	Description
0	Command Successful
4	Resolver Error
8	Syntax Error
12	Packet Allocation Error
16	Socket Error
100	Command line is too long

Examples

Example 1:

```
snmptrap trap 1.3.6.1.2.1.1.4.1 counter 132 IUCV SNMPD03
```

The example above will send a trap using an AF_IUCV socket to the SNMP agent SNMPD03 that looks like this:

```
version       = SNMPv1
community     = public
generic type  = 6 (enterpriseSpecific
oid           = 1.3.6.1.2.1.1.4.1
specific type = 6 (COUNTER)
value         = 132
```

Example 2:

```
snmptrap trape 1.3 TICKS 12 1.3.7 text "Error 2pm" ent 1.6.7 inet 9.60.2.1 comm test
```

This example will send an extended trap using an AF_INET socket to the SNMP Agent on host 9.60.2.1 that looks like this:

```
version       = SNMPv1
community     = TEst
enterprise    = 1.6.7
generic type  = 6 (enterpriseSpecific)
oid           = 1.3
specific type = 8 (TICKS)
```

```
value         = 12
oid           = 1.3.7
specific type = 9 (TEXT)
value         = Error 2pm
```

Major and Minor Error Codes and SNMP Value Types

The following are the possible major and minor error codes and variable value types that can be returned in an SNMP response or trap.

- The major error code can have one of the following values.

Value	Major Error Code
0	No error detected
1	SNMP agent reported error
2	Internally detected error

- The minor error code can have one of the following values when the major error code indicates that an SNMP agent detected an error.

Value	SNMP Agent Detected Minor Error Code
0	No error
1	Too big
2	No such name
3	Bad value
4	Read only
5	General error

- The minor error code can have one of the following values when the major error code indicates that an internal error was detected.

Value	Internal Minor Error Code
0	No error
1	Protocol error
2	Out of memory
3	No response—all retries failed
4	Some I/O error occurred
5	Illegal request
6	Unknown host specified
7	Unknown MIB variable
8	No such filter
9	Too many variables specified

- If the major error code indicates that an SNMP agent detected the error, the error index indicates the position of the first variable in error.
- The variable value type is one of the following (as specified in RFC 1155 and RFC 1156).

Value	Value Type
0	Text representation
1	Number (integer, signed)
2	Binary data string
3	Object identifier
4	Empty (no value)
5	Internet address
6	Counter (unsigned)
7	Gauge (unsigned)
8	Time ticks (1/100ths seconds)
9	Display string

Note: The binary data string is displayed in NetView as a contiguous string of hexadecimal characters (for example, X'0123' is displayed as 0123).

gethostbyname()

When a host name is specified with an SNMP request, the SNMP Query Engine looks up the IP address of that host. It uses the standard *gethostbyname*() function to perform that function. The IP address is then saved in an in-memory cache for future references. For more information about *gethostbyname()*, see *z/VM: TCP/IP Programmer's Reference*.

The cache cannot be refreshed, and if for some reason the mapping between host names and IP addresses changes, the SNMP Query Engine (the SQESERV module) has to be restarted to rebuild its cache. This is also true for a host name that was found to be nonexistent at the time of the first SNMP request, but which has been added to the name server database.

This gives a performance boost to subsequent requests for the same host.

Chapter 14. Using the Domain Name System

This chapter describes the domain name system (DNS), resolvers, and resource records. This chapter also provides descriptions of the NSLOOKUP and DIG programs used to query name servers.

Overview of the Domain Name System

The TCP/IP for z/VM DNS is comprised of only resolver application programs. For more information about these services, see "Resolvers" on page 397. Configuring these services is described in *z/VM: TCP/IP Planning and Customization*. "Domain Name Servers" on page 396 provides general information about the servers that are part of a Domain Name System.

TCP/IP applications map domain names to an internet address to identify network nodes. Mapping must be consistent across the network to ensure interoperability. The DNS provides this mapping, through network nodes called domain name servers. The DNS can provide additional information about nodes and networks, including the TCP/IP services available at a node and the location of name servers in a network.

The DNS defines a special domain called `in-addr.arpa` to translate internet addresses to domain names. An `in-addr.arpa` name is composed of the reverse octet order of an IP address concatenated with the `in-addr.arpa` string. For example, a host named `Host1` has `9.67.43.100` as an internet address. The `in-addr.arpa` domain translates the `host1` internet address `9.67.43.100` to `100.43.67.9.in-addr.arpa`.

For a complete description of the DNS, see RFC 1033, RFC 1034, and RFC 1035, which define the internet standard.

Domain Names

The DNS uses a hierarchical naming convention for naming hosts. Each host name is composed of domain labels separated by periods. Local network administrators have the authority to name local domains within an internet. Each label represents an increasingly higher domain level within an internet. The fully qualified domain name of a host connected to one of the larger internets generally has one or more subdomains.

For example:

host.subdomain.subdomain.rootdomain
or
host.subdomain.rootdomain

Domain names often reflect the hierarchy level used by network administrators to assign domain names. For example, the domain name `eng.mit.edu`. is the fully qualified domain name where `eng` is the host, `mit` is the subdomain, and `edu` is the highest level domain (root domain).

Figure 61 on page 396 is an example of the DNS used in the hierarchy naming structure across an internet.

© Copyright IBM Corp. 1987, 2016

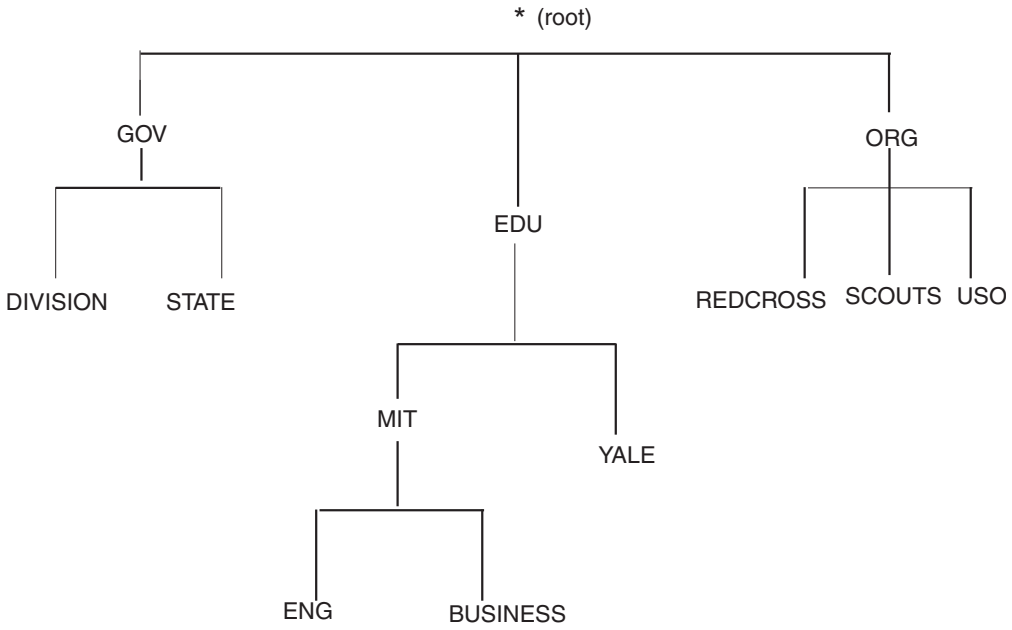


Figure 61. Hierarchical Tree

You can refer to hosts in your domain by host name only; however, a name server requires a fully qualified domain name. The local resolver appends the domain name before sending the query to the domain name server for address resolution.

Domain Name Servers

Domain name servers are designated network nodes that maintain a database of information about all nodes in a zone. The complete database is not kept by any one name server on a network. A name server has a zone of authority that is a subnetwork, or a group of subnetworks, for which the name server maintains a database. A name server is authoritative only within its zone of authority.

To minimize dependency on a particular node, the name server's database for a zone is replicated at several nodes. At least one of the nodes is designated as the primary name server. The others are secondary name servers. The zone data updates and maintenance are reflected in the primary name server. The secondary name servers update their database by contacting the primary name server at regular intervals. Both primary and secondary name servers are authoritative for a zone.

The zones of authority are arranged in a hierarchy based on the domain origin components. A special zone known as the *root* exists at the top of the domain name hierarchy in a network. The root zone contains a list of all the root servers. For example, in the internet, the root name servers store information about nodes in the root domain, and information about the delegated domains, such as `com` (commercial), `edu` (education), and `mil` (military). The root name servers store the names of name servers for each of these domains, which in turn store the names of name servers for their delegated subdomains.

TCP/IP applications contact a name server whenever it is necessary to translate a domain name into an internet address, or when information is required about a domain. The name server performs the translation if it has the necessary information. If it does not have the necessary information, the name server can contact other name servers, which in turn can contact other name servers. This

process is called a recursive query. Alternatively, a name server can simply return the address of another name server that might hold the requested information. This is called a referral response to a query. Name server implementations must support referrals, but are not required to perform recursive queries. For more information about query responses, see "Resolvers."

Some name server implementations maintain a cache of query responses sent out to other name servers on behalf of clients. This improves the processing speed for queries about domain names outside the server's zone of authority. However, responses derived from cached information are considered nonauthoritative and are flagged as such in the response.

The Network Information Center (NIC) is responsible for network and user registration, including network number, top-level domain name assignment, and `in-addr.arpa` zone assignment. For more information contact:

```
Government Systems, Inc.
Attention:  Network Information Center
14200 Park Meadow Drive,  Suite 200
Chantilly, VA  22021
1-(800)-235-3155
(internet address:  nic@nic.ddn.mil)
```

Resolvers

TCP/IP provides three programs for interactively querying a name server:
- NSLOOKUP
- DIG
- A CMS command interface (CMSRESOL)

For more information about these programs, see "NSLOOKUP—Querying Name Servers" on page 401, "DIG—Querying Name Servers" on page 421, and "CMSRESOL—Resolver and Name Server" on page 433.

Programs that query a name server are called resolvers. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for application programmers to perform queries. Under z/VM, these routines are available in the TCP/IP application programming interface (API) for each supported language.

Resolvers operate by sending query packets to a name server, either over the network or to the local name server.

A query packet contains the following fields:
- a domain name
- a query type
- a query class

"Resource Records" on page 398 lists valid query class (network class) and query type (data type) records. The name server attempts to match the three fields of the query packet to its database.

The name server can return the following query responses:

Response
Description

Authoritative
Returned from a primary or secondary name server. The name server contains all the domain data used to define the zone for the specified query.

Nonauthoritative
Returned from a cache kept by a name server. The cache does not contain the domain data used to define the zone for the specified query.

Referral
Contains the addresses of other name servers that can answer the query. A referral response is returned when a recursive query is not supported, not requested, or cannot be answered because of network connectivity.

Negative
Indicates that no records of the requested type were found for the domain name specified, if returned from an authoritative name server.

Name Error
Indicates that no resource records of any type (including wildcards) exist for the domain name specified.

Format Error
Indicates that the name server found an error in the query packet sent by the resolver.

Not-implemented
Indicates that the name server does not support the type of query requested.

Refused
Indicates that the name server refuses to perform the specified operation. For example, some root name servers limit zone transfers to a set number of IP addresses.

Data from a name server is stored and distributed in a format known as a resource record. Resource record fields are described in detail in "Resource Records." Each response from a name server can contain several resource records that can contain a variety of information. The format of a response is defined in RFC 1035, and includes the following sections:

- A question section, echoing the query for which the response is returned.
- An answer section, containing resource records matching the query.
- An additional section, containing resource records that do not match the query, but might provide useful information for the client. For example, the response to a query for the host name of a name server for a specific zone includes the internet address of that name server in the additional section.
- An authority section, containing information specific to the type of response made to the query. If a referral is returned, this section contains the domain names of name servers that could provide an authoritative answer. If a negative response is returned indicating the name does not exist, this section contains a Start Of Authority (SOA) record defining the zone of authority of the responding name server.

Resource Records

Resource records are name server database records. These records contain the following fields, in order:

Field Description

Domain name
Specifies the domain name identifying a network object. A network object can be a network, a specific node, a mailbox (for a network user's mail), or other objects addressable by the DNS.

TTL Indicates the number of seconds that a record is valid in a cache.

Network class
Specifies the network class. The allowable values are:

CHAOS
CHAOS system (obsolete)

HESIOD
Hesiod class

IN The internet (most Domain Name Systems support only the internet (IN) class)

The wildcard value ANY is defined to match any of these classes.

Data type
Indicates the type of data record. The following is a list of valid record types:

SOA Start of authority record

The SOA record is unique to a zone. This record contains the administrative details of the zone, including:

- The domain name of the name server responsible for the zone
- The mail address of the user responsible for the zone
- The serial number of the zone database, which identifies the current revision of the data
- The refresh interval, which indicates the length of time, in seconds, you must allow between the refreshing of a database from a remote name server
- The retry interval, which indicates the length of time, in seconds, you must allow before retrying a failed refresh
- The expiration TTL, which indicates the maximum time, in seconds, for records to be valid in the zone database
- The minimum TTL, which indicates the minimum time, in seconds, for records to be valid in the zone database

NS Name server record

The name server record contains the domain name of a name server for the current zone.

A Address record

The address record contains the dotted-decimal notation internet address for the domain name identifying the record.

AAAA
IPv6 Address record

The address record contains the colon hexadecimal notation internet address for the domain name identifying the record.

CNAME
Canonical name record

The canonical name record is used to provide alias or alternative name information for a domain name. The domain name specified in the first field of the record is an alternative to the canonical or real domain name specified in the data field.

HINFO
Host Information Record

This record type contains a text string specifying the CPU (central processing unit) type and operating system of a node.

MB The mailbox record (experimental)

The mailbox record contains the domain name of a host machine to receive mail for the user specified in the domain name field.

MG Mail group member record (experimental)

The mail group member record specifies the mail address of a person belonging to the mail group specified in the domain name field.

MINFO
Mailbox information record (experimental)

The mailbox information record specifies the mail addresses of the persons responsible for the mail group specified in the domain name field.

MR Mail rename name record (experimental)

The mail rename name record specifies a mailbox that is a rename of the mailbox specified in the domain name field.

MX Mail exchanger record

The mail exchanger record identifies a host able to act as a mail exchange for the domain specified in the domain name field. A mail exchange runs a mail agent that delivers or forwards mail for the domain name specified in the first field.

NULL Null resource record (experimental)

The null resource record contains any information, providing it is less than 65 535 octets in length.

PTR Domain name pointer record

The domain name pointer record is mainly used to store data for the `in-addr.arpa` and `IP6.arpa` domain, and contains the domain name referenced by an internet address.

TXT Text string record

The text string record contains descriptive text.

WKS Well-known services record

The well-known services record stores the protocol numbers of multiple services in a single record. Each of the defined TCP/IP services has a unique protocol number. For more detailed information, see RFC 1060.

For flexibility, the following wildcard query data are defined:

Type **Description**

ANY Any record type for the domain name

AXFR The query type used by secondary name servers to transfer all records in the zone (the query class is set to IN when using the AXFR query type)

MAILB
Any mailbox records for the domain name.

Data Contains information appropriate for the data type indicated in the data type field, in the format defined for that specific data type.

NSLOOKUP—Querying Name Servers

NSLOOKUP is a program for querying name servers. The NSLOOKUP program allows you to:
- Locate information about network nodes
- Examine the contents of a name server database
- Establish the accessibility of name servers

NSLOOKUP Internal State Information

The internal state information of NSLOOKUP determines the operation and results of your name server queries. The following list shows (in order) the places where the code checks for state information:
1. TCP/IP client program configuration file, TCPIP DATA
2. NSLOOKUP startup file, NSLOOKUP ENV
3. NSLOOKUP subcommands in command line mode
4. NSLOOKUP subcommands in interactive session mode

NSLOOKUP retains the last value set by any one of the items in the list. This means you can configure the internal state information of NSLOOKUP according to the last value set by any one of the items.

For more information about the TCPIP DATA file, see *z/VM: TCP/IP Planning and Customization*.

After parsing the command line, NSLOOKUP attempts to read the startup file, NSLOOKUP ENV. This file contains NSLOOKUP **set** subcommands only, which define the NSLOOKUP defaults. Enter each **set** subcommand on a separate line; blank lines are not accepted. For more information about **set** subcommands available in NSLOOKUP, see "NSLOOKUP set Subcommands" on page 409.

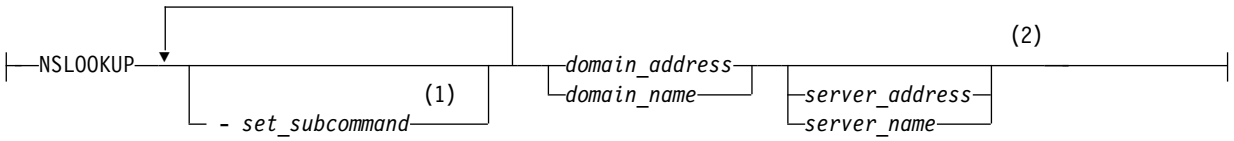
The following is an example of the contents of the NSLOOKUP ENV file:

```
set domain=powers.oz
querytype=HINFO
set norecurse
vc
```

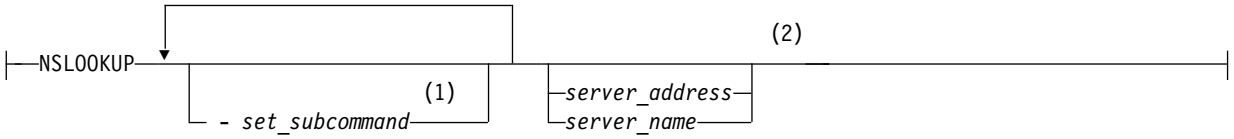
Note: Specifying **set** before the NSLOOKUP subcommands in the NSLOOKUP ENV file is optional.

NSLOOKUP Command

Using the NSLOOKUP command line mode:



Starting an NSLOOKUP interactive session:



Using the NSLOOKUP interactive session mode:

Notes:

1 Space is not allowed between the dash (-) and the **set** subcommand.

2 NSLOOKUP operands and subcommands are case sensitive and must be entered in lowercase. Operand and subcommand values and domain names are not case sensitive.

Purpose

Use the NSLOOKUP command to issue a single name server query in command line mode or multiple name server queries in interactive session mode.

Operands

set_subcommand
Specifies an NSLOOKUP **set** subcommand. For available **set** subcommands, see "NSLOOKUP set Subcommands" on page 409.

interactive_subcommand
Specifies an NSLOOKUP interactive session subcommand. For available interactive session subcommands, see "NSLOOKUP Interactive Session Subcommands" on page 404.

domain_address
Reverses the components of the address, and generates a pointer type (PTR) query to the name server for the `in-addr.arpa` or `IP6.arpa` domain mapping of the address to a domain name.

domain_name
Queries the name server for information about *domain_name* using the current query type. The default query type is A (address query).

server_address
Specifies the internet address of the name server to be queried other than the default name server. A query for the address in the `in-addr.arpa` or `IP6.arpa` domain is initially made to the default name server to map the internet address to a domain name for the server.

server_name
Directs the default name server to map *server_name* to an internet address and then uses the name server at that internet address.

Usage Notes

1. You must specify the host name or internet address of the host where the agent is running, as well as the community name to be used. The community name used for a SET request is frequently different than the community name for a GET request. You must specify the names and values of each variable to be set. RFC 1156 and RFC 1158 define the variables that you can set with read-write access. In addition, there may be enterprise-specific variables as defined by the implementer of a particular SNMP agent.
2. If the NSLOOKUP ENV file exists, the **set** subcommands are read from the file and executed before any queries are made. For more information about configuring NSLOOKUP internal state information using the NSLOOKUP ENV file, see "NSLOOKUP Internal State Information" on page 401.
3. NSLOOKUP interprets typing or syntax errors as queries. This results in a query being sent and the name server response displayed. The response is usually `Non-existent Domain`, which indicates that the server could not find a match for the query.
4. A timeout occurs if the name server is not running or is unreachable.
5. A `Non-existent Domain` error occurs if any resource record type for the specified domain name is not available from the name server.
6. A `Server Failed` error occurs when the local name server cannot communicate with the remote name server.
7. NSLOOKUP performs a query for the domain specified. The query requests all information about the domain using the current class and query (resource record) type. You can specify a server other than the current server to perform the domain name resolution.
8. In interactive session mode, an initial query is made to the selected name server to verify that the server is accessible. All subsequent interactive queries are sent to that server, unless you specify another server using the **set server** or **set lserver** subcommands.
9. You can make a query by entering the domain name of the node or subnetwork for which information is required and define the data type of information to be retrieved using the **set querytype** subcommand. You can define only one type of resource record for a domain name in a single query, unless the wildcard query type of **any** has been set.
10. Queries processed by NSLOOKUP that specify an address can give unexpected results. If the current query type is address (A) or domain name pointer (PTR), NSLOOKUP generates a PTR type query for the specified address in the `in-addr.arpa` or `IP6.arpa` domain. This returns PTR records, which define the host name for the specified address. If the current query type is neither of these two types, a query is performed using the current query type, with the domain name specified as the address given.
11. NSLOOKUP does not issue a query for a domain name if the name is unqualified and is the same as one of the defined options.

12. A name server often requires a fully qualified domain name for queries. However, NSLOOKUP allows the specification of a default subnetwork domain using the **set domain** subcommand, with the initial default obtained from the TCPIP DATA file. When **defname** is enabled, using the **set defname** subcommand, the default domain name specified by **set domain** is appended to all unqualified domain names. For example, if the default domain name is `fourex.oz` and **defname** is enabled, a query for the name `toolah` automatically generates a query packet containing the domain name `toolah.fourex.oz`.
13. The domain name or address for a query can be followed by the domain name or internet address of a name server to contact for the query. If this is not specified, the current name server is used. For example, `toolah wurrup.fourex.oz` queries the name server on `wurrup.fourex.oz` for information about the node `toolah`.
14. When specifying domain names that include periods, the trailing period (indicating a fully qualified domain name) is optional. NSLOOKUP strips the trailing period if it is present. If you are specifying a root domain, the domain name must have two trailing periods. For example, specify `mynode..` when the node `mynode` is in the root domain.

Examples

When you specify NSLOOKUP **set** subcommands in command line mode, do not use **set** preceding the subcommand. For example, to specify a name server (NS) type record lookup for the domain name `fourex.oz`, enter the following on the command line:

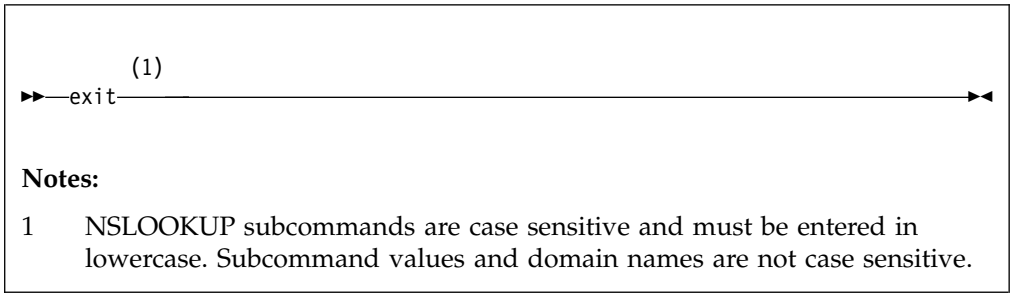
```
nslookup -querytype=ns fourex.oz
```

The option **-querytype=ns** is a **set** subcommand, but **set** is omitted from the command. For more information about using **set** subcommands, see "NSLOOKUP set Subcommands" on page 409.

NSLOOKUP Interactive Session Subcommands

The NSLOOKUP subcommands explained in this topic can be used during an NSLOOKUP interactive session. For more information on starting and using an NSLOOKUP interactive session, see "NSLOOKUP Command" on page 401.

exit



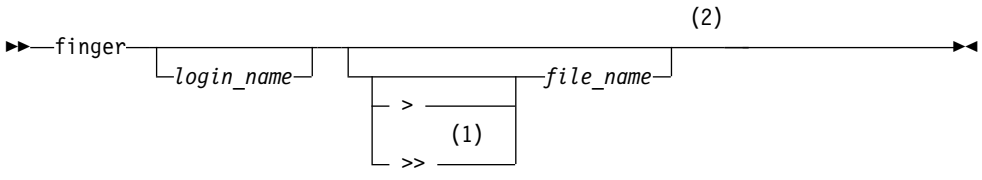
Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **exit** interactive session subcommand to exit from interactive session mode.

finger



Notes:

1 There must be at least one space before and after the > and >> operands.

2 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **finger** interactive session subcommand to extract information from the finger server of the host found in the last address query.

Operands

login_name
The login name of the user for which to return information about. The login name is case sensitive and must be specified in the same case (upper or lower) as that used by the host.

> Specifies that the output will overwrite the contents of the file specified by *file_name*.

>> Specifies that the output will be appended to the contents of the file specified by *file_name*.

file_name
The file name of the file where output will be written for later viewing.

Usage Notes

1. By default, a list of users logged in on the host last queried is returned. Specify a user's login name to return information about a specific user.
2. The **finger** subcommand expects that the finger server is operating on the host found in the last address query. An error message is displayed if the finger server is not operating or if the host cannot be reached.
3. If a current host is not defined, querying the name server defines that name server to be the current host for a subsequent finger operation.
4. An error message is displayed if the preceding subcommand was not a successful address query or finger operation.

help / ?

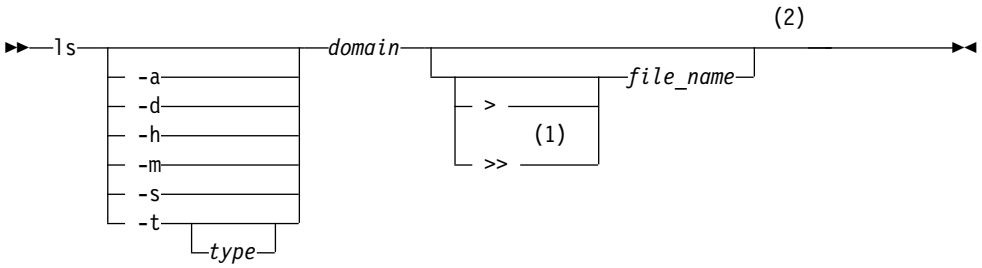
Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **help** or **?** interactive session subcommand to display a brief summary of subcommands.

ls



Notes:

1 There must be at least one space before and after the > and >> operands.

2 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **ls** interactive session subcommand to list the information available for the specified domain.

Operands

-a Retrieves the CNAME resource record.

-d Retrieves the ALL resource record.

-h Retrieves the HINFO resource record.

-m Retrieves the MX resource record.

-s Retrieves the WKS resource record.

-t

type
Retrieves the resource record type specified by *type*. If no record type is specified, the current default type is used. For more information on valid resource record field types, see Data type.

domain
The domain name zone to be queried.

> Specifies that the output will overwrite the contents of the file specified by *file_name*.

>> Specifies that the output will be appended to the contents of the file specified by *file_name*.

file_name
The file name of the file where output will be written for later viewing.

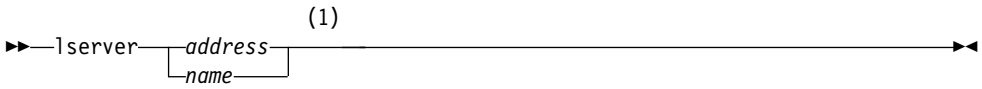
Usage Notes

1. The **ls** subcommand works correctly only if the name server is configured using the one-answer zone transfer format. If the name server sends multiple answer records, the DIG command can be used to correctly process the zone transfer. Use the DIG command AXFR query type option to perform the zone transfer. For example:

   ```
   DIG @nameserver domain AXFR
   ```

   where *nameserver* is the host name or internet address of your name server and *domain* is the domain name.
2. The **ls** subcommand creates a virtual circuit (TCP/IP connection) with the current name server to service the request. An error message is displayed if the virtual circuit cannot be established.
3. An error message is displayed and no information is listed if the domain name specified by *domain* refers to a host.
4. If the output is being written to a file, a number sign (#) is displayed as every 50 lines of output are written to the file to indicate the command is still processing.

lserver



Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **lserver** interactive session subcommand to change the current name server.

Operands

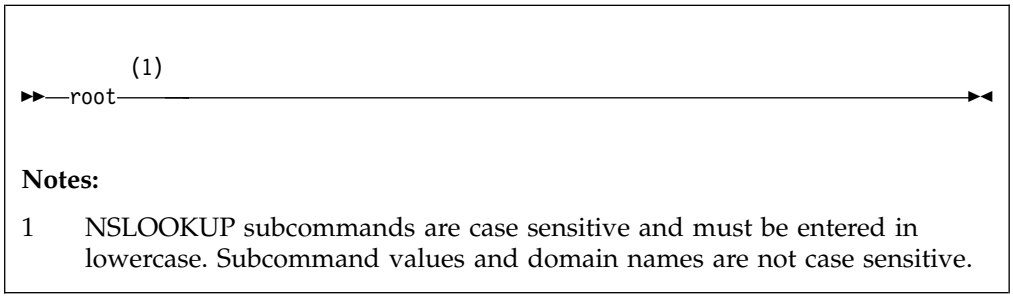
address

name
The new name server internet address (*address*) or host name (*name*).

Usage Notes

1. If a host name is specified, the internet address is resolved using the initial name server defined at command invocation. An error will occur when the host name cannot be mapped to an internet address.
2. The **lserver** subcommand does not verify that you can contact the name server at the specified address; it simply changes a local variable storing the address of the default name server.

root



Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

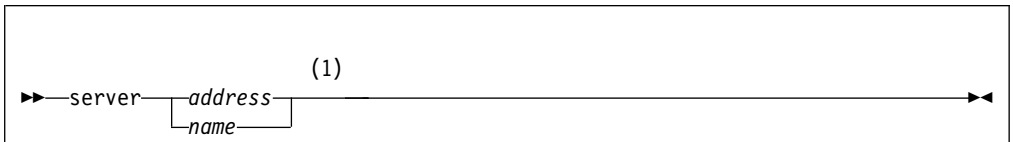
Purpose

Use the **root** interactive session subcommand to change the current name server to the root name server.

Usage Notes

1. The default root name server is `ns.nic.ddn.mil`. To change the default root name server, use the **set root** subcommand (which is equivalent to **lserver** ***name***). For more information on using the **set root** subcommand, see "NSLOOKUP set Subcommands" on page 409.
2. An error will occur if the root name server internet address cannot be resolved.
3. The **root** subcommand does not verify that you can contact the name server at the defined address; it simply changes a local variable storing the address of the default root name server.

server



Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **server** interactive session subcommand to change the current name server.

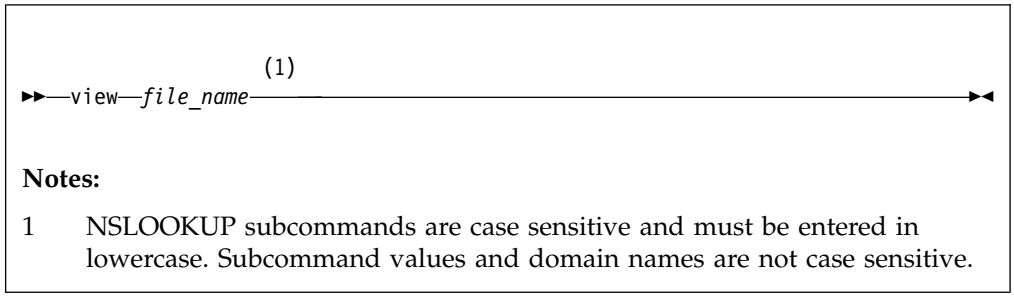
Operands

address
name
The new name server internet address (*address*) or host name (*name*).

Usage Notes

1. If a host name is specified, the internet address is resolved using the current name server. An error will occur when the host name cannot be mapped to an internet address.
2. The **server** subcommand does not verify that you can contact the name server at the specified address; it simply changes a local variable storing the address of the default name server.

view



Notes:

1 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **view** interactive session subcommand to sort and list the contents of a file one screen at a time.

Operands

file_name
The file name of the file of which the contents will be sorted and listed. An error occurs if the file does not exists.

NSLOOKUP set Subcommands

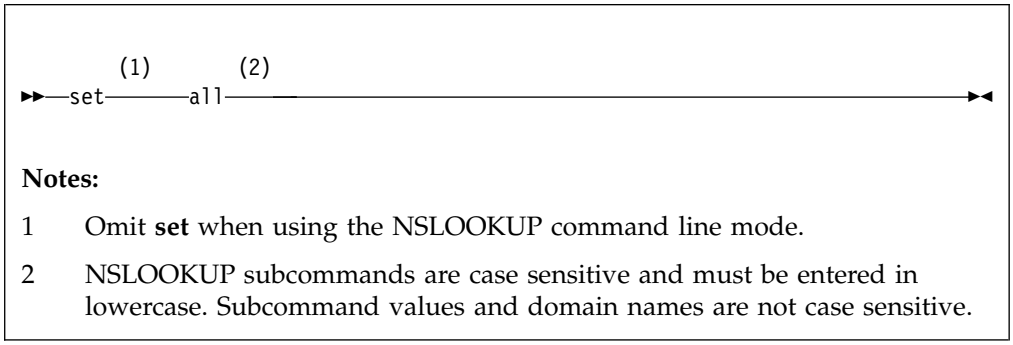
Internal state information affects the operation and results of your queries. You can change the internal state information maintained by NSLOOKUP using the NSLOOKUP **set** subcommands. Some internal state information is initially retrieved from the TCPIP DATA file. For more information about the TCPIP DATA file, see *z/VM: TCP/IP Planning and Customization*. For more information about configuring NSLOOKUP internal state information, see "NSLOOKUP Internal State Information" on page 401.

The NSLOOKUP **set** subcommands explained in this topic can be used on the command line, during an NSLOOKUP interactive session, or in the NSLOOKUP ENV file. When NSLOOKUP **set** subcommands are specified in a command line query, you must omit "set"; preceding the subcommand. If the NSLOOKUP **set**

subcommands are specified in an interactive session query, “set” is required. In the NSLOOKUP ENV file, specifying “set” is optional.

For more information on using an NSLOOKUP command line session or starting and using an NSLOOKUP interactive session, see “NSLOOKUP Command” on page 401.

set all



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

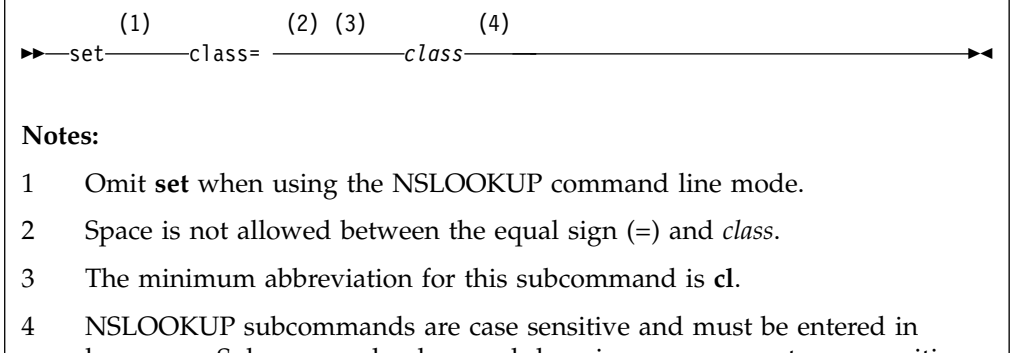
Purpose

Use the **set all** subcommand to display the current values of the internal state variables.

Usage Notes

1. The **set all** subcommand does not alter the internal state of NSLOOKUP.

set class



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 Space is not allowed between the equal sign (=) and *class*.

3 The minimum abbreviation for this subcommand is **cl**.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set class** subcommand to change the class of information returned by a query.

Operands

class
The class mnemonic for the class of information to be returned by a query. For more information about classes recognized by NSLOOKUP, see the resource record field, Network class.

set d2 / set nod2

Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

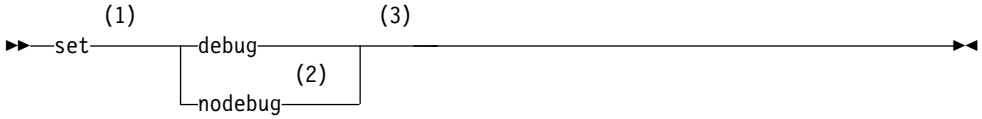
Use the **set d2** subcommand to enable extra debugging mode.

Use the **set nod2** subcommand to disable extra debugging mode.

Usage Notes

- Extra debugging mode is disabled by default.
- Debug mode is also enabled when using the **set d2** subcommand.

set debug / set nodebug



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for these subcommands are **deb** and **nodeb**.

3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

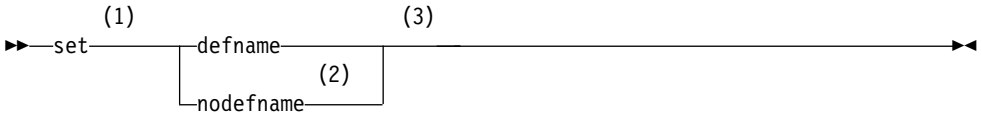
Use the **set debug** subcommand to enabled debugging mode.

Use the **set nodebug** subcommand to disable debugging mode.

Usage Notes

1. Debug mode is disabled by default.
2. Extra debugging mode is also disabled when using the **set nodebug** subcommand.

set defname / set nodefname



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for these subcommands are **def** and **nodef**.

3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

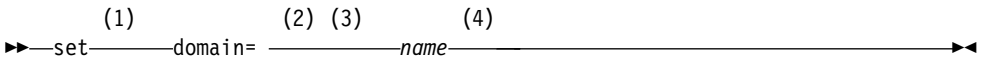
Use the **set defname** subcommand to enable appending the default domain name to an unqualified domain name in a query.

Use the **set nodefname** subcommand to disable appending the default domain name to an unqualified domain name in a query. That is, the specified domain name is passed to the name server without modification.

Usage Notes

1. The default domain name is initially obtained from the TCPIP DATA file, but can be changed using the **set domain** subcommand. For more information on using the **set domain** subcommand, see "set domain."
2. Appending the default domain name to an unqualified domain name is enabled by default.

set domain



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **do**.

3 Space is not allowed between the equal sign (=) and *name*.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

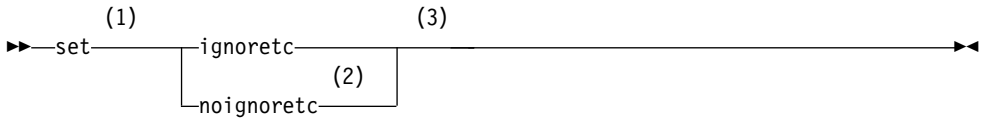
Purpose

Use the **set domain** subcommand to change the default domain name. In addition, the search list is updated to contain only the domain name specified.

Operands

name
The domain name to be used as the default domain name. The value specified is not verified. The default domain name is initially obtained from the TCPIP DATA file.

set ignoretc / set noignoretc



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for these subcommands are **ig** and **noig**.

3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set ignoretc** subcommand to direct NSLOOKUP to ignore the truncation indicator when it is set in a name server response.

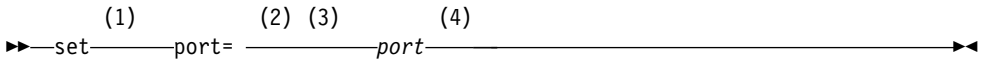
Use the **set noignoretc** subcommand to direct NSLOOKUP to automatically try the query again, using a TCP connection, when a response is sent with the truncation indicator set.

The truncation indicator, in the response header, is set by the name server when a complete query response did not fit into a single UDP packet and has been truncated.

Usage Notes

1. NSLOOKUP does not handle responses greater than 512 characters in length. Responses greater than 512 characters are truncated and the internal truncation flag is set. This condition is revealed only when debugging mode is enabled. For more information on enabling debugging mode, see “set debug / set nodebug” on page 411.
2. The truncation indicator is ignored by default.

set port



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **po**.

3 Space is not allowed between the equal sign (=) and *port*.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

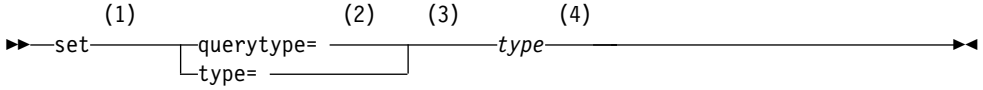
Purpose

Use the **set port** subcommand to change the default port used to contact a name server.

Operands

port
The port number to be used as the default port for contacting a name server. The default port used by NSLOOKUP for contacting a name server is port 53.

set querytype / set type



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **q**.

3 Space is not allowed between the equal sign (=) and *type*.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set querytype** or **set type** subcommand to change the type of information to be returned by a name server query.

Operands

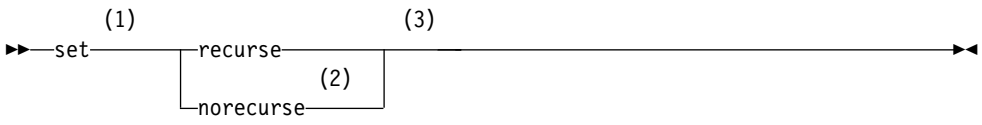
type
The query type to be used when querying a name server. The default query

type is "A" (address information). For more information about query types recognized by NSLOOKUP, see the resource record field, Data type.

Usage Notes

1. Only one query type can be specified.
2. NSLOOKUP cannot generate type NULL queries. In this case, NSLOOKUP displays the number of bytes returned in the NULL record.
3. Global queries that return all resource records for a specific domain name are specified by the wildcard value ANY.

set recurse / set norecurse



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for these subcommands are **rec** and **norec**.

3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

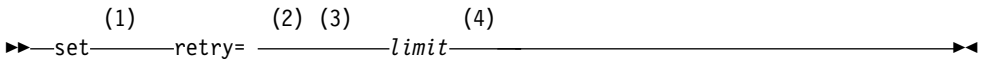
Use the **set recurse** subcommand to request a recursive query when querying a name server.

Use the **set norecurse** subcommand to request a non-recursive query when querying a name server.

Usage Notes

1. Recursive queries are returned by default.

set retry



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **ret**.

3 Space is not allowed between the equal sign (=) and *limit*.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set retry** subcommand to change the number of times a request is resent.

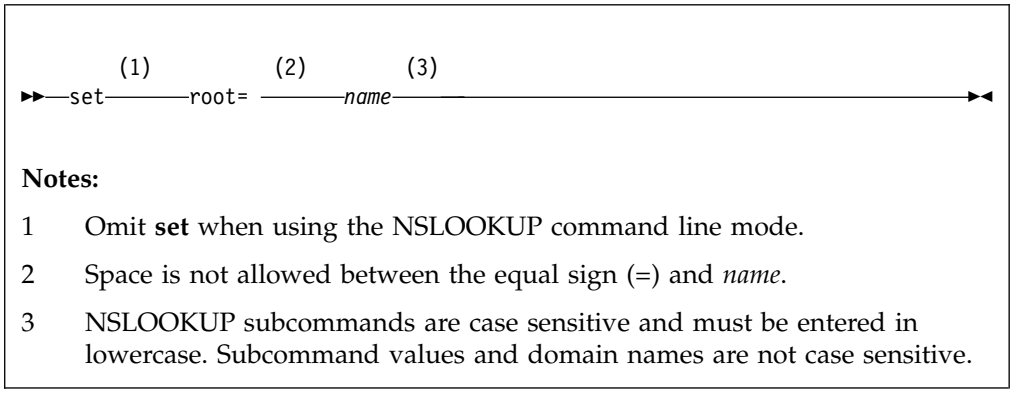
Operands

limit
The number of repeat attempts to send a request to a name server. That is, when a request is sent and the timeout period expires, the request is resent until the specified limit has been exceeded. The default limit is initially obtained from the TCPIP DATA file.

Usage Notes

1. Setting the limit to zero disables NSLOOKUP from contacting the name server. The result is the following message: `no response from server`.
2. The NSLOOKUP retry algorithm uses both a limit and a timeout period. The timeout period for each resent request is twice the timeout period of the previous request.

set root



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 Space is not allowed between the equal sign (=) and *name*.

3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

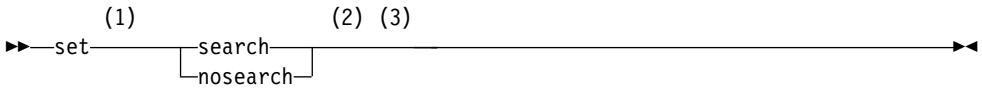
Purpose

Use the **set root** subcommand to change the domain name of the root server.

Operands

name
The domain name of the new root server. The default root server is `ns.nic.ddn.mil`.

set search / set nosearch



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for these subcommands are **sea** and **nosea**.

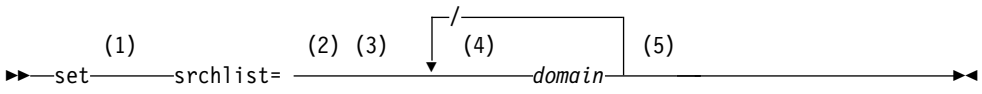
3 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set search** subcommand to enable the use of a search list.

Use the **set nosearch** subcommand to disable the use of a search list.

set srchlist



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **srchl**.

3 Space is not allowed between the equal sign (=) and *domain*.

4 A maximum of three domains can be specified. Each domain is separated by a forward slash (/).

5 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set srchlist** subcommand to specify up to three domain names to be appended to unqualified host names when attempting to resolve the host name.

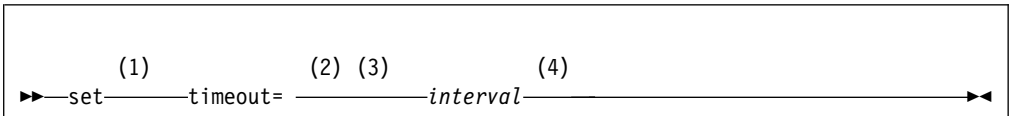
Operands

domain
Specifies the domain name to be appended to unqualified host names. Each domain name specified is tried, in turn, until a match is found.

Usage Notes

1. This subcommand changes the default domain to be the first domain name specified in the search list.

set timeout



Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 The minimum abbreviation for this subcommand is **t**.

3 Space is not allowed between the equal sign (=) and *interval*.

4 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set timeout** subcommand to change the number of seconds to wait before a request times out.

Operands

interval
Number of seconds to wait before a request times out. The default interval is initially obtained from the TCPIP DATA file.

set vc / set novc

Notes:

1 Omit **set** when using the NSLOOKUP command line mode.

2 NSLOOKUP subcommands are case sensitive and must be entered in lowercase. Subcommand values and domain names are not case sensitive.

Purpose

Use the **set vc** subcommand to enable the use of a virtual circuit (TCP connection) to transport queries to the name server.

Use the **set novc** subcommand to disable the use of a virtual circuit (TCP connection) to transport queries to the name server and instead use datagrams (UDP).

Usage Notes

1. The default setting is retrieved from the TCPIP DATA file. If no entry is found, the default is to use datagrams (UDP).

NSLOOKUP Examples

This section contains examples of NSLOOKUP command line mode queries, and interactive session mode queries using the various options available for NSLOOKUP commands.

In Figure 62, the router, `wurrup`, has two internet addresses and there are two name servers, `wurrup` being the primary name server. This network is described by a single zone in the domain naming hierarchy stored in the name servers. The domain name is fourex.oz.

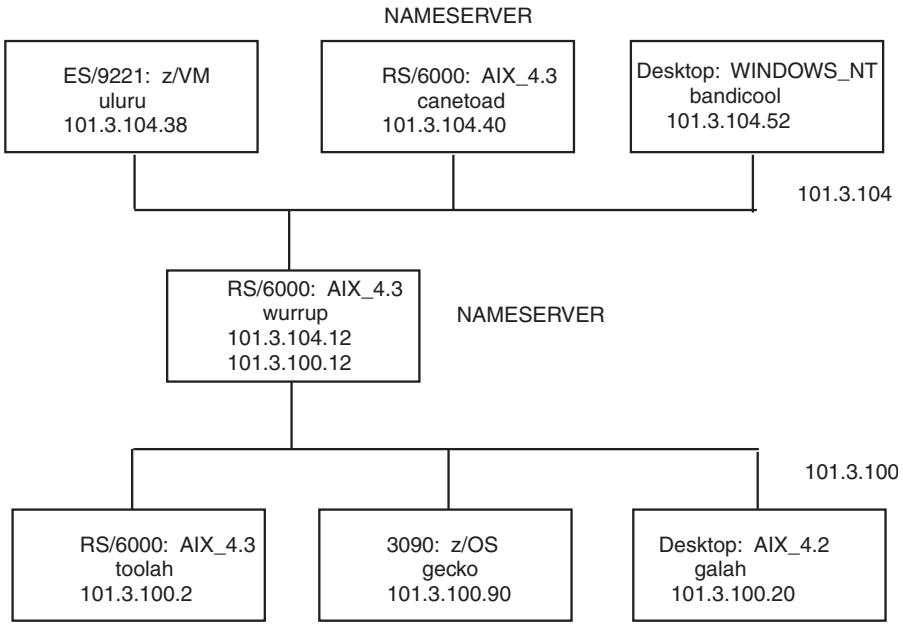


Figure 62. A TCP/IP Network

The following are examples of how to use NSLOOKUP to extract information from a name server. The queries are executed from the z/VM host `uluru` at IP address *101.3.104.38* on the network described in Figure 62.

The following examples are command line mode queries.

1. To make a simple address query:

```
   User:   nslookup toolah.fourex.oz wurrup.fourex.oz
 System:   Server:  wurrup
Address:   101.3.104.12

           Name:   toolah.fourex.oz
           Address:   101.3.100.2
```

2. To specify a name server (NS) type record lookup:

```
   User:   nslookup -query=ns fourex.oz
 System:   Server:  canetoad
Address:   101.3.104.40

           fourex.oz  nameserver = wurrup.fourex.oz
           fourex.oz  nameserver = canetoad.fourex.oz
```

```
            wurrup.fourex.oz   internet address = 101.3.100.12
            wurrup.fourex.oz   internet address = 101.3.104.12
            canetoad.fourex.oz internet address = 101.3.104.40
```

The following command places NSLOOKUP in interactive session mode with wurrup as the default server.

```
   User:   nslookup - wurrup
 System:   Default Server:  wurrup
Address:   101.3.104.12
```

The following examples are all in the interactive session mode initiated in the preceding example.

1. Show the default flag settings:

```
     User:   set all
   System:   >; Default Server:  wurrup
  Address:   101.3.104.12

             Set options:
             nodebug         defname         nosearch        recurse
             nod2            novc            noignoretc      port=53
             querytype=A     class=IN        timeout=60      retry=1
             root=ns.nic.ddn.mil
             domain=FOUREX.OZ
             srchlist=FOUREX.OZ
```

2. Perform a simple address query:

```
     User:   toolah
   System:   >; Server:  wurrup
  Address:   101.3.104.12

             Name:    toolah.FOUREX.OZ
             Address:  101.3.100.2
```

3. Set the query record type to HINFO, and perform another query:

```
     User:   set q=HINFO
             toolah
   System:   >; >; Server:  wurrup
  Address:   101.3.104.12

             toolah.FOUREX.OZ   CPU = RS6000     OS = AIX3.2
```

4. Find out the name servers available for a domain:

```
     User:   set q=NS
             fourex.oz
   System:   >; >; Server:  wurrup
  Address:   101.3.104.12

             fourex.oz  nameserver = wurrup.fourex.oz
             fourex.oz  nameserver = canetoad.fourex.oz
             wurrup.fourex.oz   internet address = 101.3.100.12
             wurrup.fourex.oz   internet address = 101.3.104.12
             canetoad.fourex.oz internet address = 101.3.104.40
```

5. Change the current server from wurrup to canetoad and make more queries:

```
     User:   server canetoad
   System:   >; Default Server:  canetoad.fourex.oz
  Address:   101.3.104.40

     User:   set q=A
             gecko
   System:   >; Server:  canetoad.fourex.oz
  Address:   101.3.104.40

             Name:    gecko.fourex.oz
             Address:  101.3.100.90
```

6. Enable debugging and execute a simple query to see the result, and then disable debugging:

```
   User:   set deb
           wurrup
 System:   >; >; Server:  canetoad.FOUREX.OZ
Address:   101.3.104.40

           res_mkquery(0, wurrup.FOUREX.OZ, 1, 1)
           ------------
           Got answer:
              HEADER:
                 opcode = QUERY, id = 7, rcode = NOERROR
                 header flags:  response, auth. answer, want recursion,
                 recursion avail
                 questions = 1, answers = 2, authority records = 0,
                 additional = 0

              QUESTIONS:
                 wurrup.FOUREX.OZ, type = A, class = IN
              ANSWERS:
              ->; wurrup.FOUREX.OZ
                 internet address = 101.3.104.12
                 ttl = 9999999 (115 days 17 hours 46 mins 39 secs)
              ->; wurrup.FOUREX.OZ
                 internet address = 101.3.100.12
                 ttl = 9999999 (115 days 17 hours 46 mins 39 secs)

           ------------
           Name:    wurrup.FOUREX.OZ
           Addresses:  101.3.104.12, 101.3.100.12
   User:   set nodeb
```

7. Find all addresses in the `fourex.oz` domain using the **ls** subcommand:

```
   User:   ls fourex.oz
 System:   >; canetoad.FOUREX.OZ
           fourex.oz                    server = wurrup.fourex.oz
           wurrup                        101.3.100.12
           wurrup                        101.3.104.12
           fourex.oz                    server = canetoad.fourex.oz
           canetoad                      101.3.104.40
           gecko                         101.3.100.90
           wurrup                        101.3.100.12
           wurrup                        101.3.104.12
           galah                         101.3.100.20
           bandicoot                     101.3.104.52
           toolah                        101.3.100.2
           canetoad                      101.3.104.40
           loopback                      127.0.0.1
           uluru                         101.3.104.38
```

8. Find all aliases in the `fourex.oz` domain, then exit from NSLOOKUP interactive session mode:

```
   User:   ls -a fourex.oz
 System:   >; canetoad.FOUREX.OZ
           localhost                     loopback.fourex.oz
           infoserver                    wurrup.fourex.oz
           pabxserver                    wurrup.fourex.oz
   User:   exit
```

DIG—Querying Name Servers

DIG is a program for querying domain name servers. The DIG program allows you to:

- Exercise name servers
- Gather large volumes of domain name information

- Execute simple domain name queries

Note: Currently, DIG supports IPv4 only.

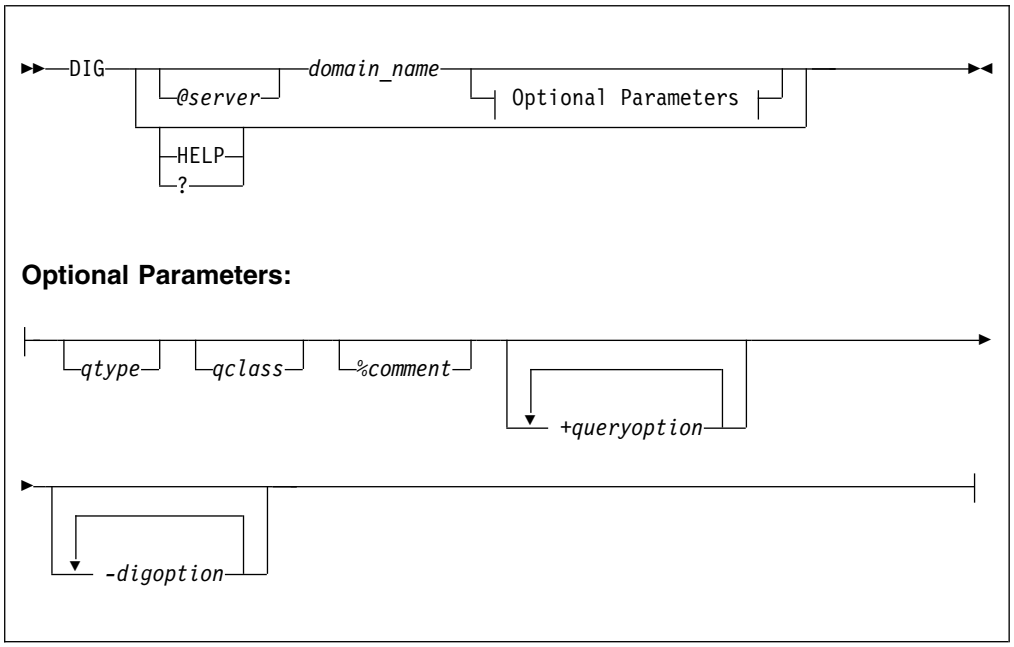
DIG Internal State Information

The internal state information of DIG determines the operation and results of your name server queries. You can configure the internal state information of DIG using the following methods, listed in order of precedence:
- TCP/IP client program configuration file, TCPIP DATA
- DIG startup file, DIG ENV
- Query options on the command line or in a batch file

The DIG ENV file contains a list of query option defaults. This list is initialized from the DIG ENV file when DIG is invoked. The default values in DIG ENV are used for all queries unless overridden by query flags on the command line. The defaults can be reset during a batch run by using the `-envset` flag on a batch file line. For more information about the query options available for DIG, see "Query Options" on page 424.

The DIG ENV file is created and updated using the `-envsav` option, which writes the current defaults out to the file after parsing the query options on the command line. The `-envsav` option specified on the command line and the existing default values are saved in the DIG ENV file as the default environment for future invocations of DIG. The DIG ENV file is not reread when the environment is updated during batch queries and the `-envsav` flag has no effect on subsequent queries in a batch file. The DIG ENV file is written in nontext format and cannot be viewed or edited.

DIG Command



Note: The *queryoption* and *digoption* parameters are case sensitive and must be entered in lowercase. Domain names, query types, query classes, and the values associated with *queryoption* and *digoption* parameters are not case sensitive.

Purpose

Use the DIG command to query a domain name server in command line mode or batch mode.

Operands

server
Specifies the domain name or internet address of the name server to contact for the query. The default is the name server specified in the TCPIP DATA file.

If a domain name is specified, DIG uses the resolver library routines provided in the TCP/IP for VM programming interface to map the name to an internet address.

HELP
? Provides help information about the DIG command. You cannot place the HELP operand on the DIG command line with other operands; they will be ignored.

domain_name
Specifies the name of the domain for which information is requested. If the domain name does not exist in the default domain specified in the TCPIP DATA file, a fully qualified domain name must be specified.

qtype
Specifies the type of query to be performed. DIG does not support MAILA, MD, MF, and NULL query types. The wildcard query types are ANY, MAILB, and AXFR.

For more information about valid query types, see the resource record field Data type.

If the *qtype* option is omitted, the default query type is A (an address query).

qclass
Specifies which network class to request in the query. DIG recognizes only the IN, CHAOS, HESIOD, and ANY network classes.

For descriptions of these classes, see the resource record field Network class.

comment
Enables you to include comments in a DIG command. Any characters following the percent (%) character up to the next space character (space or end-of-record) are ignored by DIG. This option is useful in batch files for annotating a command.

For example, using a dotted-decimal notation internet address rather than a domain name removes any overhead associated with address mapping; however, this makes the command less readable. Therefore, in a batch file you can include the domain name as a comment for readability.

queryoption
Interprets the string following the plus sign (+) character as a query option. Query options have the format:

parameter[=*value*]

and are a superset of the set subcommand options for NSLOOKUP. See “Query Options” on page 424 for the available query options.

digoption
Interprets the string following the minus sign (-) as a DIG option. The DIG options are either a parameter or a single character followed by a parameter. See "DiG Options" on page 426 for the available DIG options.

Query Options

[no]aaonly
Specifies whether to accept only authoritative responses or all responses to queries. The default is `noaaonly` (accept all responses).

[no]addit
Specifies whether to print the additional section of the response. The additional section contains resource records that have not been explicitly requested, but could be useful. For more information about this option, see RFC 1035. The default is `addit` (print the additional section).

[no]answer
Specifies whether to print the answer section of the response. The answer section contains the set of all resource records from the name server database that satisfy the query. The default is `answer` (print the answer section).

[no]cl
Specifies whether to print network class information for each of the resource records returned. The default is `nocl` (do not print network class information).

[no]cmd
Specifies whether to echo the parsed options. The default is `cmd` (echo parsed options).

[no]d2
Specifies whether to print the details of each query sent out to the network, including send time-stamp and time-out time-stamp. When a server does not respond within the time-out period, DIG either sends the query to another server, or resends the query to the original server. The details of the query are visible when `d2` is set. The default is `nod2` (do not print query details).

[no]debug
Directs DIG to print additional error messages. The default is `debug` (print additional error messages).

[no]defname
Specifies whether to append the default domain name to all unqualified domain names in a query. The default domain name is set by specifying the `+domain=`*name* option. If the nodefname option is set, the domain name specified is passed to the server without modification. The default is `defname` (append the default domain name).

domain=*name*
Sets the default domain name to *name*. Initially, the default domain name is obtained from the TCPIP DATA file. The validity of *name* is not verified. If the `defname` option is set, the domain name specified in *name* is appended to all unqualified domain names before the queries are sent to the name server.

[no]Header
Specifies whether to print the header line containing the operation code, returned status, and query identifier of each response. This option is distinct from the `header` option. The default is `Header` (print the header).

[no]header
Specifies whether to print the query flags of each response. The query flags are defined in RFC 1035. The default is `header` (print the query flags).

[no]ignore
Specifies whether to report truncation errors. Truncation errors occur when a response is too long for a single datagram. Specifying `ignore` directs DIG to ignore truncation errors. The default is `noignore` (report any truncation errors).

[no]ko
Specifies whether to keep the virtual circuit open for queries in batch mode only. This option has no effect when used on the command line or when datagrams are used to transport queries (see the `novc` option). The default is `noko` (create a new virtual circuit for each batch query).

pfand=*number*
Performs a bitwise AND of the current print flags with the value specified in *number*. The number can be octal, decimal, or hexadecimal.

pfdef
Sets the print flags to their default values.

pfmin
Sets the print flags to the minimum default values. This option specifies that minimal information should be printed for each response.

pfor=*number*
Performs a bitwise OR of the current print flags with the value specified in *number*. The number can be octal, decimal, or hexadecimal.

pfset=*number*
Sets the print flags to the value specified in *number*. The number can be octal, decimal, or hexadecimal.

[no]qr
Specifies whether to print the outgoing query. The outgoing query consists of a header, question section and empty answer, additional, and authoritative sections. See RFC 1035 for more information about outgoing queries. The default is `noqr` (do not print the outgoing query).

[no]ques
Specifies whether to print the question section of a response. The question section contains the original query. The default is `ques` (print the question section).

[no]recurse
Specifies whether to request a recursive query when querying a name server. The `norecurse` option specifies that a recursive query is not requested. The default is `recurse`.

[no]reply
Specifies whether to print the response from the name server. When this option is disabled, other print flags that affect printing of the name server response are ignored and no sections of the response are printed. The default is `reply` (print the response).

retry=*limit*
Specifies the number of times a request is resent. When a request is sent and the time-out period expires for a response, the request is resent until the value specified in *limit* has been exceeded. The value specified in *limit* determines the number of attempts made to contact the name server. The default value for *limit* is retrieved from the TCPIP DATA file.

Setting *limit* to zero disables DIG from contacting the name server. The result is an error message `no response from server`.

The retry algorithm for DIG uses both the *limit* value and the time-out period. Each time a request is resent, the time-out period for the request is twice the time-out period used for the last attempt.

[no]sort
Specifies whether to sort resource records before printing. When performing the wildcard query type AXFR, the sort option is ignored if the name server being contacted sends DNS packets containing multiple answer records. The default is `nosort` (do not sort resource records).

[no]stats
Specifies whether to print the query statistics including round trip time, time and date of query, size of query and response packets, and name of server used. The default is `stats` (print the query statistics).

timeout=*time_out_value*
Specifies the number of seconds to wait before timing out of a request. The default time out value is retrieved from the TCPIP DATA file.

[no]ttlid
Specifies whether to print the TTL (time to live) for each resource record in a response. The default is `ttlid` (print time to live).

[no]vc
Specifies whether to use a virtual circuit (TCP connection) to transport queries to the name server or datagrams (UDP). The default is retrieved from the TCPIP DATA file. If no entry is found, the default is `novc` (use datagrams)

DiG Options

c *query_class*
Specifies that the command line query or batch query retrieves resource records having the given network class. The *qclass* parameter, described under "Operands" on page 423, can also be used to specify the query class. In addition to the mnemonics, this option also accepts the equivalent numeric value that defines the class.

envsav
Directs DIG to save the environment specified on the current command line in the DIG ENV file. This DIG ENV file initializes the default environment each time DIG is invoked.

The DIG environment is described in "DIG Internal State Information" on page 422.

envset
Directs DIG to set the default environment, specified on the current line in the batch file. This default environment remains in effect for all subsequent queries in the batch file, or until the next line in the batch file containing the `-envset` option is reached. This option is valid for batch mode only.

The DIG environment is described in "DIG Internal State Information" on page 422.

f *file*
Specifies a file for DIG batch mode queries. The batch file contains a list of queries that are to be executed in order. The keyword `DIG` is not used when specifying queries in a batch file. Lines beginning with a number sign (#) or semicolon (;) in the first column are comment lines, and blank lines are

ignored. Options that are specified on the original command line are in effect for all queries in the batch file unless explicitly overwritten. The following is an example of a batch file.

```
# A comment
; more comments
wurrup any in +noH =noqu -c IN

toolah +pfmin
```

P Directs DIG to execute a ping command for response time comparison after receiving a query response. The last three lines of output from the following command are printed after the query returns:

```
PING -s server_name 56 3
```

p *port*
Use the port number given when contacting the name server. The Domain Name System is a TCP/IP well-known service and has been allocated port 53. DIG uses 53 by default, but this option allows you to override the port assignment.

[no]stick
Restores the default environment, before processing each line of a batch file. This flag is valid for batch mode only. If you set the `stick` option, queries in the batch file are not affected by the options specified for preceding queries in the file.

The DIG environment is described in "DIG Internal State Information" on page 422.

If you set the `nostick` option, the query option specified on the current line in the batch file remains in effect until the option is overridden by a subsequent query. The result of each query in the batch file depends on the preceding queries. The default is `nostick`.

T *seconds*
Specifies the wait time between successive queries when operating in batch mode. The default wait time is 0 (do not wait).

t *query_type*
Specifies that the query retrieves resource records having the given resource record type. The *qtype* parameter under "Operands" on page 423 can also be used to specify the query type. In addition to the mnemonics, this parameter also accepts the equivalent numeric value that defines the type.

x *dotted_decimal_notation_address*
Simplifies the specification of a query for the `in-addr.arpa` domain. Normally these queries are made by specifying a query type of `PTR` for `nn.nn.nn.nn.in-addr.arpa`, where the four `nn` components are replaced by the dotted-decimal notation internet address components in reverse order. This option allows you to make this query by simply specifying the dotted-decimal notation internet address.

For example, the domain name corresponding to internet address `101.3.100.2` is found by a query for the domain name `2.100.3.101.in-addr.arpa`. You can use `DIG -x 101.3.100.2` instead of reversing the address and appending `in-addr.arpa`.

Usage Notes

1. You can use DIG in command line mode, where all options are specified on the invoking command line, or in batch mode, where a group of queries are placed

in a file and executed by a single invocation of DIG. DIG provides a large number of options for controlling queries and screen output, including most of the functions of NSLOOKUP.

2. You can create a file for batch mode queries using the `-f file` option. The file contains complete queries, one for each line, that are executed in a single invocation of DIG. The keyword `DIG` is not used when specifying queries in a batch file. Blank lines are ignored, and lines beginning with a number sign (#) or a semicolon (;) in the first column are comment lines.
3. Options specified on the initial command line are in effect for all queries in the batch file unless explicitly overridden. Several options are provided exclusively for use within batch files, giving greater control over the operation of DIG.
4. Some internal state information is retrieved from the TCPIP DATA file. See *z/VM: TCP/IP Planning and Customization* for more information about the TCPIP DATA file.

DIG Examples

The following section provides examples of how to use DIG to extract information from a name server. In Figure 62 on page 419, the router `wurrup` has two internet addresses, and there are two name servers, `wurrup` being the primary name server. This network is described by a single zone in the domain naming hierarchy stored in the name servers. In the examples, all queries are issued from the z/VM `uluru` system.

1. Create a default environment (default options) that gives minimal output from subsequent DIG commands:

```
System:   Ready
  User:   dig wurrup +noqu +noH +nohe +nost +noad +noau +nost +nocl
          +nottl -envsav
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

The following queries show which part of the response output is controlled by each of the output control options. Each example enables or disables query options for tailoring output. The `wurrup.fourex.oz` domain name is used for the following queries.

1. Set the query type to `ns`, the query class to `in`, and print the additional section of the output:

```
System:   Ready
  User:   dig fourex.oz ns in +ad
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 3
          ;; ANSWERS:
          fourex.oz NS      wurrup.fourex.oz
          fourex.oz NS      canetoad.fourex.oz
          ;; ADDITIONAL RECORDS:
          wurrup.fourex.oz  A       101.3.100.12
          wurrup.fourex.oz  A       101.3.104.12
          canetoad.fourex.oz        A       101.3.104.40
```

2. Set the query type to `ns`, the query class to `in`, print the additional section of the output, but do not print the answer section:

```
System:   Ready
  User:   dig fourex.oz ns in +addit +noanswer
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 3
          ;; ADDITIONAL RECORDS:
          wurrup.fourex.oz  A       101.3.100.12
          wurrup.fourex.oz  A       101.3.104.12
          canetoad.fourex.oz        A       101.3.104.40
```

3. Query a nonexistent domain and print the authoritative section of the output:

```
System:   Ready
  User:   dig noname +author
System:   ;; ->>HEADER<<- opcode: QUERY , status: NXDOMAIN, id: 3
          ; Ques: 1, Ans: 0, Auth: 1, Addit: 0
          ;; AUTHORITY RECORDS:
          fourex.oz SOA     wurrup.fourex.oz  adb.wurrup.fourex.oz (
                                    10003   ;serial
                                    3600    ;refresh
                                    300     ;retry
                                    3600000 ;expire
                                    86400 ) ;minim
```

4. Use the default query options:

```
System:   Ready
  User:   dig wurrup
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

5. Print the network class information:

```
System:   Ready
  User:   dig wurrup +cl
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  IN   A   101.3.104.12
          wurrup.FOUREX.OZ.  IN   A   101.3.100.12
```

6. Echo the input query:

```
System:   Ready
  User:   dig wurrup +cmd
System:   ; <<>> DiG 2.0 <<>> wurrup +cmd
          ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

7. Print the question section of the output:

```
System:   Ready
  User:   dig wurrup +qu
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.FOUREX.OZ, type = A, class = IN

          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

8. Do not print the header line:

```
System:   Ready
  User:   dig wurrup +noH
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

9. Print the query flags:

```
System:   Ready
  User:   dig wurrup +he
System:   ;; flags: qr aa rd ra ; Ques: 1, Ans: 2, Auth: 0, Addit:
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

10. Print the question section and the outgoing query:

```
System:   Ready
  User:   dig wurrup +qu +qr
System:   ; Ques: 1, Ans: 0, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.FOUREX.OZ, type = A, class = IN

          ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.FOUREX.OZ, type = A, class = IN

          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

11. Print the query statistics including round-trip time:

```
System:   Ready
  User:   dig fourex.oz ns in +stats
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 3
          ;; ANSWERS:
          fourex.oz NS      wurrup.fourex.oz
          fourex.oz NS      canetoad.fourex.oz
          ;; Sent 1 pkts, answer found in time: 37 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:06:40 1993
          ;; MSG SIZE  sent: 24  rcvd: 116
```

12. Print the TTL for each resource record:

```
System:   Ready
  User:   dig fourex.oz ns in +ttlid
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 3
          ;; ANSWERS:
          fourex.oz 9999999 NS      wurrup.fourex.oz
          fourex.oz 9999999 NS      canetoad.fourex.oz
```

13. Enable extra debugging mode:

```
System:   Ready
  User:   dig wurrup +d2
System:   ;; res_mkquery(0, wurrup, 1, 1)
          ;; Querying server (# 1) address = 101.3.104.40
          ;; id = 3 - sending now: 4044656426 msec
          ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

The following examples show how options control the use and value of the default domain.

1. Do not append the default domain name to unqualified domain names and print the question section of the response:

```
System:   Ready
  User:   dig wurrup +nodefname +qu
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup, type = A, class = IN

          ;; ANSWERS:
          wurrup.fourex.oz  A       101.3.104.12
          wurrup.fourex.oz  A       101.3.100.12
```

2. Set the default domain name to `fourexpd` and print the question section of the response:

```
System:   Ready
  User:   dig wurrup +do=fourexpd +qu
System:   ;; ->>HEADER<<- opcode: QUERY , status: SERVFAIL, id: 3
```

```
          ; Ques: 1, Ans: 0, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.fourexpd, type = A, class = IN
```

3. Set the query type to `ns`, the query class to `in`, and sort the output:

```
System:   Ready
  User:   dig fourex.oz ns in +sort
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 3
          ;; ANSWERS:
          fourex.oz NS      canetoad.fourex.oz
          fourex.oz NS      wurrup.fourex.oz
```

4. Query the domain at the address `101.3.100.20` and print the question section of the response:

```
System:   Ready
  User:   dig -x 101.3.100.20 +qu
System:   ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;    20.200.9.192.in-addr.arpa, type = ANY, class = IN

          ;; ANSWERS:
          20.200.9.192.in-addr.arpa.      PTR     galah.
```

5. Retrieve resource records with a network class of `ANY` and print the question section of the response:

```
System:   Ready
  User:   dig wurrup -c any +qu
System:   ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.FOUREX.OZ, type = A, class = ANY

          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
```

6. Retrieve resource records with a query type of `ANY` and print the question section of the response:

```
System:   Ready
  User:   dig wurrup -t any +qu
System:   ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
          ;; QUESTIONS:
          ;;      wurrup.FOUREX.OZ, type = ANY, class = IN

          ;; ANSWERS:
          wurrup.FOUREX.OZ.  A       101.3.104.12
          wurrup.FOUREX.OZ.  A       101.3.100.12
          wurrup.FOUREX.OZ.  HINFO   RS6000 AIX3.2
```

The batch file, `test.digbat` used for this example is shown below. The default environment has been removed by discarding the DIG ENV file. The DiG command is omitted for all entries in the file.

Note the effect of the `-envset` and `-stick` options on the output.

```
wurrup any in +noH +nohe +noqu +noad +noau -envset -stick
wurrup any in
toolah a in +d2
toolah a in
toolah a in +d2 -nostick
toolah a in
toolah a in +nod2
toolah a in
```

1. Specify the batch file `test.digbat`:

```
System:   Ready
  User:   dig -f test.digbat

System:   ; <<>> DiG 2.0 <<>> dig wurrup any in +noH +nohe +noqu +noad
          +noau -envset -st k
          ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  9999999 A       101.3.104.12
          wurrup.FOUREX.OZ.  9999999 A       101.3.100.12
          wurrup.FOUREX.OZ.  86400   HINFO   RS6000 AIX3.2
          ;; Sent 1 pkts, answer found in time: 20 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 95

System:   ; <<>> DiG 2.0 <<>> dig wurrup any in
          ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
          ;; ANSWERS:
          wurrup.FOUREX.OZ.  9999999 A       101.3.104.12
          wurrup.FOUREX.OZ.  9999999 A       101.3.100.12
          wurrup.FOUREX.OZ.  86400   HINFO   RS6000 AIX3.2
          ;; Sent 1 pkts, answer found in time: 112 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 95

System:   ; <<>> DiG 2.0 <<>> dig toolah a in +d2
          ;; res_mkquery(0, toolah, 1, 1)
          ;; Querying server (# 1) address = 101.3.104.40
          ;; id = 3 - sending now: 4046124888 msec
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A       101.3.100.2
          ;; Sent 1 pkts, answer found in time: 210 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 47

System:   ; <<>> DiG 2.0 <<>> dig toolah a in
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A       101.3.100.2
          ;; Sent 1 pkts, answer found in time: 270 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 47

System:   ; <<>> DiG 2.0 <<>> dig toolah a in +d2 -nostick
          ;; res_mkquery(0, toolah, 1, 1)
          ;; Querying server (# 1) address = 101.3.104.40
          ;; id = 3 - sending now: 4046125037 msec
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A    101.3.100.2
          ;; Sent 1 pkts, answer found in time: 360 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 47

System:   ; <<>> DiG 2.0 <<>> dig toolah a in
          ;; res_mkquery(0, toolah, 1, 1)
          ;; Querying server (# 1) address = 101.3.104.40
          ;; id = 5 - sending now: 4046125101 msec
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A       101.3.100.2
          ;; Sent 1 pkts, answer found in time: 24 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 47
```

```
System:   ; <<>> DiG 2.0 <<>> dig toolah a in +nod2
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A       101.3.100.2
          ;; Sent 1 pkts, answer found in time: 19 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:57 1993
          ;; MSG SIZE  sent: 31  rcvd: 47
System:   ; <<>> DiG 2.0 <<>> dig toolah a in
          ; Ques: 1, Ans: 1, Auth: 0, Addit: 0
          ;; ANSWERS:
          toolah.FOUREX.OZ.  9999999 A       101.3.100.2
          ;; Sent 1 pkts, answer found in time: 26 msec
          ;; FROM: FOUREXVM1 to SERVER: default -- 101.3.104.40
          ;; WHEN: Tue Mar 16 11:15:58 1993
          ;; MSG SIZE  sent: 31  rcvd: 47
```

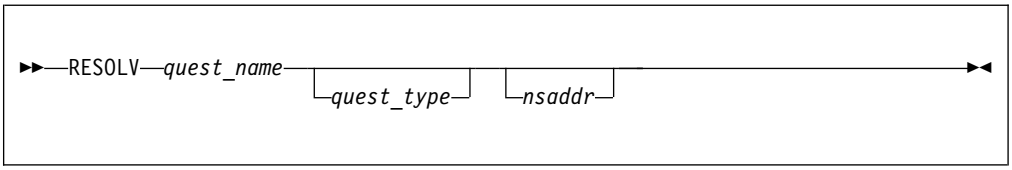
CMSRESOL—Resolver and Name Server

The resolver is a program or subroutine that obtains information from a name server or site tables to convert host names into internet addresses.

The name server is used for cross-referencing a name to its corresponding internet address. The name server uses an internal database or other name servers as its source of information.

Note: Currently, CMSRESOL supports IPv4 only.

RESOLV Command—Interface to the CMSRESOL MODULE



Purpose

The RESOLV EXEC provides a sample EXEC interface to the CMSRESOL MODULE to perform standard Name Server queries. The sample provided has limited capability, but can be modified as needed using a VMSES/E local modification.

Operands

quest_name
Specifies the name of the resource that is the target of the query. This is a required parameter.

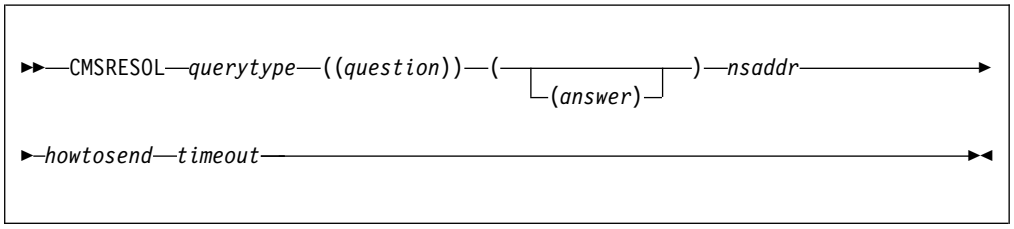
quest_type
Specifies the query question type (Qtype). If omitted, the Qtype defaults to A (host address query type).

The valid values for this parameter are listed in the header file CMOLVER.H. They are also listed in RFC 883.

nsaddr
Specifies the internet address of the name server from which you want a

response. If omitted, the first NSINTERADDR value defined in the TCPIP DATA file is used; if no such value is found, the VM TCP/IP loopback address (127.0.0.1) is used.

CMSRESOL Command—Interface to the Resolver



Purpose

The CMSRESOL program can send queries to the name server using TCP or UDP. The CMSRESOL program sends the query and receives the results by the communication method defined on the CMSRESOL command. The results are placed on the program stack.

Operands

querytype
Specifies QUERY or IQUERY.
QUERY
Specifies the standard query
IQUERY
Specifies the inverse query

question
Specifies the form of Qname, Qtype, and Qclass.

answer
Specifies the form of Name, Type, Class, TTL, and Data.

nsaddr
Specifies the internet address of the name server. For example: `101.3.104.40.`

howtosend
Indicates the form of the query. The form can be:
DATAGRAM
Specifies the use of UDP datagrams
CONN
Specifies the use of TCP streams

timeout
Indicates the number of seconds to wait when attempting to open a connection to the name server.

The *question*, *answer*, and *authority* sections are returned as separate lines in the program stack (system data queue). They can be extracted from the queue with the REXX PULL instruction. The CMSRESOL command results in one of the following return codes:

Return Code	Description
0	No error condition

1 Format error
2 Name server failure
3 Domain name does not exist
4 Not implemented
5 Refused by name server
20 Parameter error
21 Error in communicating with name server
22 Software error
100 Command line is too long

Note:

1. The name server uses an internal buffer of 4096 bytes to transfer data. If this limit is exceeded, a name server failure-error condition is returned.
2. Each line in the program stack can hold as many as 255 characters. Data in excess of this limit is wrapped into additional lines in the program stack.
3. You must allow at least one space between adjacent sets of parentheses for these queries to work.

Types of Queries

This section contains examples of standard, inverse, in-addr.arpa domain, and database queries, as well as examples of queries outside your domain.

Note: The figure of 86 400, which appears in the program stack, is a typical TTL value of 24 hours, expressed in seconds.

The Standard Query

A standard query provides the *question*, and the *answer* is returned. An example of a standard query is:

```
CMSRESOL QUERY (  ( RALPH.YKT.IBM.COM A IN )  )  (  )  (  ) 9.0.0.1
               DATAGRAM 60
```

The results of the query are placed on the program stack. An example of the format of the data on the stack resulting from this standard query is:

```
(  ( RALPH.YKT.IBM.COM A IN )  )
(  ( RALPH.YKT.IBM.COM A IN 86400 192.5.5.5 )  )
(  )
(  )
```

The Inverse Query

An inverse query provides the *answer*, and the *question* is returned. An example of an inverse query using a TCP virtual circuit and the resulting stack contents is:

```
CMSRESOL IQUERY (  )  (  ( X.YKT.IBM.COM A IN 0 192.5.5.5 )  )  (  )
                9.0.0.1 CONN 60
```

The stack contents are:

```
(  ( RALPH.YKT.IBM.COM A IN )  )
(  ( RALPH.YKT.IBM.COM A IN 86400 192.5.5.5 )  )
(  )
(  )
```

Querying the in-addr.arpa Domain

Only local, authoritative data is returned from a name server in response to an inverse query. Recursion is not available, because an inverse query does not supply

the domain origin in the data field. To resolve an internet address to a domain name, a special domain exists called in-addr.arpa. The in-addr.arpa domain contains the internet address of the name field in reverse order, suffixed with in-addr.arpa. To resolve the internet address to a domain name, do a standard query supplying the internet address, in reverse order, suffixed with in-addr.arpa in the name field.

An example of this type of query is:

```
CMSRESOL QUERY (  ( 126.43.67.9.IN-ADDR.ARPA PTR IN )  )  (  )  (  ) 9.0.0.1
               DATAGRAM 45
```

The results of the query are placed on the program stack. An example of the format of the data on the stack resulting from this query is:

```
(  ( 126.43.67.9.IN-ADDR.ARPA PTR IN )  )
(  ( 126.43.67.9.IN-ADDR.ARPA PTR IN  86400 VMX.RALEIGH.IBM.COM )  )
(  )
(  )
```

Querying Outside Your Domain

The following is a sample of a query outside your domain:

```
CMSRESOL QUERY  (  ( ALPHA.UNKNOWN.COM A IN )  )  (  )  (  )
                9.0.0.1 DATAGRAM 30
```

The results of the query are placed on the program stack. An example of the format of the data on the stack resulting from this query is:

```
(  ( ALPHA.UNKNOWN.COM A IN )  )
(  )
(  ( UNKNOWN.COM NS IN 86400 NAMESERVER.UNKNOWN.COM )  )
(  ( NAMESERVER.UNKNOWN.COM A IN 86400 128.1.2.3 )  )
```

Chapter 15. Using Translation Tables

TCP/IP uses translation tables to convert transmitted data between EBCDIC and ASCII. Because the meanings of the terms "EBCDIC" and "ASCII" depend on the particular operating system and the national language (English, French, etc.) being used on a particular system, TCP/IP provides many different translation tables to meet the diverse needs of z/VM users.

In addition to the more than 200 translations provided by IBM, you can create custom tables to meet your specific requirements.

The following sections provide the information you need to understand what translation tables are, how they are used by TCP/IP applications, and how you can create your own custom translations.

Character Sets and Code Pages

When you display or print a document, you see a collection of characters or symbols. A group of characters or symbols taken together and treated as a single entity is called a **character set**. A character set may contain hundreds or even thousands of characters.

In a Single-Byte Character Set (SBCS), one 8-bit byte is used to represent a single character. This means there are only 256 possible bit patterns or **code points** available to represent a character. All Western languages can be represented by an SBCS character set.

A Double-Byte Character Set (DBCS) uses *two* bytes to represent a single character, providing a theoretical maximum of 65536 characters. In practice, DBCS character sets contain far fewer than 65536 characters. Eastern languages such as Japanese Kanji, Korean Hangeul, and traditional Chinese require a DBCS character set.

A collection of all of 256 (for SBCS) or 65536 (for DBCS) code points and their corresponding individual character assignments are called a **code page**.

While it is true that always using a universal DBCS character set such as Unicode would eliminate the need to perform EBCDIC-ASCII translation, most of the operating systems and standard TCP/IP application protocols in use today were developed before the advent of DBCS. As a consequence, every country or common geographic region developed its own country-specific SBCS code page, particularly in the EBCDIC environment. Characters were deleted, added, and their order changed.

Consequently, it is necessary to understand and manage the use of code pages. To assist in that effort, IBM has assigned a unique number to many of the EBCDIC and ASCII code pages you will use. The specific code page translations provided with TCP/IP are listed in Table 40 on page 440. Facilities are provided so that you may supplement the translations provided by IBM with your own.

The TCP/IP translation tables convert data from one code page to another, so the table you choose depends on the code pages being used by the systems involved and your knowledge of how a file was created.

© Copyright IBM Corp. 1987, 2016

It is important to recognize that changing the default translation table for servers such as FTP and NFS can corrupt data in a file if that file is uploaded and downloaded using different translation tables. (This does not apply to binary transfers, of course.)

TCP/IP Translation Table Files

TCP/IP translation tables are machine-readable binary files that are usually kept on the TCP/IP user disk, TCPMAINT 592.

Most of these files are provided by IBM, and others may be created by compiling SBCS or DBCS translation table source files using the CONVXLAT command, described in "Converting Translation Tables to Binary" on page 446.

Table 38. Translation Table Files

Character Set	Language	Source File Type	Table File Type
SBCS	Any	TCPXLATE	TCPXLBIN
DBCS	Japanese Kanji	TCPKJLAT	TCPKJBIN
DBCS	Korean Hangeul	TCPHGLAT	TCPHGBIN
DBCS	Traditional Chinese	TCPCHLAT	TCPCHBIN

Note that the file types for the different languages are different. There can be up to four translation table that have the same file name – one for all SBCS languages, and one for each of the three DBCS languages.

SBCS tables contain translations for one pair of code pages. DBCS tables may contain multiple translations.

To modify an IBM-provided translation table:
1. Modify the source file as required
2. Run the CONVXLAT program, specifying the modified source as input
3. Copy the resulting TCP*xx*BIN file to the TCP/IP user disk, TCPMAINT 592. Translation tables made available to servers should also be made available to clients.

To create a new translation table, first copy an existing source file and then follow the procedure outlined above.

SBCS translation tables may be read by CMS applications using the DTCXLATE CSL routine. For more information on DTCXLATE, see the *z/VM: CMS Callable Services Reference*.

Translation Table Search Order

Most TCP/IP client and server programs provide a way for the you to change the translation table that is used. This is done using either client command line options, server initialization parameters, or configuration file statements. For example, the CMS FTP client provides a **TRANSLATE** option that you can use to provide the name of a translation table.

If no such option or configuration statement is provided, the clients and servers will use a *preferred* translation table that is specific to a particular client or server. If the preferred table cannot be found, the common *standard* translation will be used.

The standard translation is loaded from STANDARD TCP*xx*BIN, if it is available, or from an equivalent that is compiled into the program.

Table 39 shows the option or configuration statement that may be used to provide the name of a translation table to be used by each client or server. Any program not listed can be assumed to use STANDARD.

Table 39. Preferred Translation Tables

Program	Option	Preferred Translation Table
SMTP Server	None[1]	SMTP
SMTP Client (SENDFILE)	TRANSLATE table_name[2,3]	STANDARD
FTP Server	SITE TRANSLATE *table_name*[2,4]	SRVRFTP
FTP Client	TRANSLATE *table_name*[2]	FTP
UFT Client (SENDFILE)	TRANSLATE *table_name*[2]	STANDARD
UFT Server	TRANSLATE *table_name*[2]	STANDARD
LPR Client	TRANSLATE *table_name*[2]	LPR
NFS Server	XLATE=*table_name*[2]	VMNFS
REXECD Server	None	REXECD
TELNET Client	TRANSLATE *table_name*[2]	TELNET
TELNET Server (line mode)	None	STLINMOD

Note[1]: For SMTP, an additional translation table may be specified for 8-bit MIME support. The *z/VM: TCP/IP Planning and Customization* contains more information in the "8BITMIME Statement" section.

Note[2]: *table_name* is the file name of a TCP/IP translation table.

Note[3]: This applies only when the MIME option is specified on the SENDFILE command.

Note[4]: SITE TRANSLATE is a command, issued by the FTP client, that tells the FTP server which translation table to use for the current session. For more information, see "SITE" on page 93.

If a table is explicitly referenced by a program option or configuration statement, the program will display an error message and stop if the translation table file cannot be found or loaded.

The file type of the translation table depends on whether any DBCS features are used. If Kanji translation is requested, the file type is TCPKJBIN. If Korean translation is requested, the file type is TCPHGBIN. For traditional Chinese, the file type is TCPCHBIN.

The *z/VM: TCP/IP User's Guide* contains information on the TRANSLATE option for the TCP/IP clients, and the *z/VM: TCP/IP Planning and Customization* contains information for the servers. See the *z/VM: CMS Commands and Utilities Reference* for information about the SENDFILE command.

Special Telnet Requirements

Telnet Client

The Telnet client requires a translation table that is different from the default table, STANDARD. The preferred translation table is provided by IBM as TELNET TCPXLBIN. If this file is not found, however, the default table will be used.

Country-specific translation tables for the Telnet application are provided. These tables have the file type TELXLATE. You must rename the selected file to TELNET TCPXLATE before it is converted to binary using the CONVXLAT command. The resulting TELNET TCPXLBIN should then be copied to TCPMAINT 592, replacing the IBM version.

Note: You cannot use the Telnet translation tables to change the LineFeed (X'0A') character.

Telnet Server

For line mode Telnet sessions, translation is performed by the Telnet server using the STANDARD translation table. If this table does not meet your needs, you can create an STLINMOD TCPXLATE table, convert it to binary using CONVXLAT, and copy the resulting STLINMOD TCPXLBIN file to the TCP/IP customization disk, TCPMAINT 198.

IBM-Supplied Translation Tables

In order to meet the translation needs of users and installations worldwide, more than 200 translation tables are provided with TCP/IP for your use. Some tables already exist in binary form; others require conversion using the CONVXLAT command, as described in "Converting Translation Tables to Binary" on page 446.

In order to make an IBM-supplied translation table the default translation table for a particular client or server program, store a copy of that translation table (in binary form) under the preferred translation table name for that program, as specified in Table 39 on page 439.

Table 40. IBM Translation Tables

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
United States and Canada	0037*rrrr*	TCPXLATE	37	*rrrr*
Austria and Germany	0273*rrrr*	TCPXLATE	273	*rrrr*
Denmark and Norway	0277*rrrr*	TCPXLATE	277	*rrrr*
Finland and Sweden	0278*rrrr*	TCPXLATE	278	*rrrr*
Italy	0280*rrrr*	TCPXLATE	280	*rrrr*
Spain and Spanish-speaking Latin America	0284*rrrr*	TCPXLATE	284	*rrrr*
United Kingdom	0285*rrrr*	TCPXLATE	285	*rrrr*
France	0297*rrrr*	TCPXLATE	297	*rrrr*
International	0500*rrrr*	TCPXLATE	500	*rrrr*
Iceland	0871*rrrr*	TCPXLATE	871	*rrrr*
ISO 8859-15	0924*rrrr*	TCPXLATE	924	*rrrr*
OpenExtensions (POSIX)	1047*rrrr*	TCPXLATE	1047	*rrrr*
United States and Canada (Euro)	1140*rrrr*	TCPXLATE	1140	*rrrr*
Austria and Germany (Euro)	1141*rrrr*	TCPXLATE	1141	*rrrr*
Denmark and Norway (Euro)	1142*rrrr*	TCPXLATE	1142	*rrrr*
Finland and Sweden (Euro)	1143*rrrr*	TCPXLATE	1143	*rrrr*
Italy (Euro)	1144*rrrr*	TCPXLATE	1144	*rrrr*

Table 40. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
Spain and Spanish-speaking Latin America (Euro)	1145*rrrr*	TCPXLATE	1145	*rrrr*
United Kingdom (Euro)	1146*rrrr*	TCPXLATE	1146	*rrrr*
France (Euro)	1147*rrrr*	TCPXLATE	1147	*rrrr*
International (Euro)	1148*rrrr*	TCPXLATE	1148	*rrrr*
Iceland (Euro)	1149*rrrr*	TCPXLATE	1149	*rrrr*
OpenExtensions	1047*rrrr*	TCPXLATE	1047	*rrrr*
ISO 8859-15 (EBCDIC)	0924*rrrr*	TCPXLATE	924	*rrrr*
ISO 8859-15 (ASCII)	*hhhh*0923	TCPXLATE	*hhhh*	923
OS/2	*hhhh*0850	TCPXLATE	*hhhh*	850
OS/2 (Euro)	*hhhh*0858	TCPXLATE	*hhhh*	858
ISO 8859-1 (ASCII)	*hhhh*0819	TCPXLATE	*hhhh*	819
Microsoft Windows	*hhhh*1252	TCPXLATE	*hhhh*	1252
Austria and Germany	AUSGER	TCPXLATE	273	850
Belgium	BELGIAN	TCPXLATE	500	850
Canada	CANADIAN	TCPXLATE	37	850
Denmark and Norway	DANNOR	TCPXLATE	277	850
Netherlands	DUTCH	TCPXLATE	37	850
Finland and Sweden	FINSWED	TCPXLATE	278	850
France	FRENCH	TCPXLATE	297	850
Italy	ITALIAN	TCPXLATE	280	850
Japan	JAPANESE	TCPXLATE	281	850
OpenExtensions	POSIX	TCPXLATE	1047	819
Portugal	PORTUGUE	TCPXLATE	37	850
Spain and Spanish-speaking Latin America	SPANISH	TCPXLATE	284	850
Switzerland (French)	SWISFREN	TCPXLATE	500	850
Switzerland (German)	SWISGERM	TCPXLATE	500	850
United Kingdom	UK	TCPXLATE	285	850
United States	US	TCPXLATE	37	850
Standard (SBCS)	STANDARD	TCPXLATE	EBCDIC	ASCII
Standard Japanese Kanji	STANDARD	TCPKJLAT		
JIS X0208 1978			300	X0208
JIS X0208 1983			300	X0208
Shift JIS X0208			300	X0208
Extended Unix Code			300	EUC
IBM			300	300

Table 40. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
Standard Korean Hangeul	STANDARD	TCPHGLAT		
KSC 5601 SBCS			833	1088
KSC 5601 DBCS			834	951
Hangeul SBCS			833	891
Hangeul DBCS			834	926
Standard Traditional Chinese	STANDARD	TCPCHLAT		
Traditional Chinese SBCS			037	904
Traditional Chinese DBCS			835	927

Note: In this table *hhhh* and *rrrr* represent four-digit host and remote code page numbers, respectively.

Note:

1. STANDARD TCPXLBIN is a 7-bit non-reversible translation table. For inbound data, all ASCII characters with values in the range X'80'-X'FF' will have the same translation as values X'00'-X'7F'. The high-order bit of each ASCII byte is ignored and is assumed to be zero. For example, ASCII X'B1' and X'31' will both be converted to EBCDIC X'F1'. For outbound data, EBCDIC control characters that do not have ASCII equivalents are converted to ASCII X'1A'. STANDARD should be used in situations where a client or server is known to treat the high-order bit in each byte as a parity bit.
2. All other SBCS translation tables provide a unique, one-to-one mapping of all 256 code points.
3. All tables translate ASCII LineFeed (LF, X'0A') to and from EBCDIC LF (X'25'), except for POSIX and 1047*rrrr*, which use EBCDIC NewLine (NL, X'15') instead.
4. Host code pages 924 and 1140-1149 include translations for the euro currency symbol.

Customizing SBCS Translation Tables

All SBCS translation table files contain two tables. The first table is used to translate from ASCII to EBCDIC. The second table is used to translate from EBCDIC to ASCII.

To read the translation tables, find the row for the first hex digit (**1**) and the column for the second hex digit (**2**). The point where the row and column intersect is the translation value.

For example, to find the EBCDIC translation for the ASCII character X'A7', find row A0 (**3**) and column 07 (**4**) in the following example. The point where row A0 and column 07 intersect shows a value of X'7D', so the ASCII character X'A7' will be translated to a X'7D' in EBCDIC. To customize the translation table, alter the translate value where the row and column intersect to the new value.

```
;
; ASCII-to-EBCDIC table
;                    4                        2
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;                                                 1
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 00 ;
```

```
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 10 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; 20 ;
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; 30 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; 40 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; 50 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; 60 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; 70 ;
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 80 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 90 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; A0 ; 3
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; B0 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; C0 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; D0 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; E0 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; F0 ;
;
; EBCDIC-to-ASCII table
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
  00 01 02 03 1A 09 1A 7F 1A 1A 1A 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 1A 1A 08 1A 18 19 1A 1A 1C 1D 1E 1F    ; 10 ;
  1A 1A 1C 1A 1A 0A 17 1B 1A 1A 1A 1A 1A 05 06 07    ; 20 ;
  1A 1A 16 1A 1A 1E 1A 04 1A 1A 1A 1A 14 15 1A 1A    ; 30 ;
  20 A6 E1 80 EB 90 9F E2 AB 8B 9B 2E 3C 28 2B 7C    ; 40 ;
  26 A9 AA 9C DB A5 99 E3 A8 9E 21 24 2A 29 3B 5E    ; 50 ;
  2D 2F DF DC 9A DD DE 98 9D AC BA 2C 25 5F 3E 3F    ; 60 ;
  D7 88 94 B0 B1 B2 FC D6 FB 60 3A 23 40 27 3D 22    ; 70 ;
  F8 61 62 63 64 65 66 67 68 69 96 A4 F3 AF AE C5    ; 80 ;
  8C 6A 6B 6C 6D 6E 6F 70 71 72 97 87 CE 93 F1 FE    ; 90 ;
  C8 7E 73 74 75 76 77 78 79 7A EF C0 DA 5B F2 F9    ; A0 ;
  B5 B6 FD B7 B8 B9 E6 BB BC BD 8D D9 BF 5D D8 C4    ; B0 ;
  7B 41 42 43 44 45 46 47 48 49 CB CA BE E8 EC ED    ; C0 ;
  7D 4A 4B 4C 4D 4E 4F 50 51 52 A1 AD F5 F4 A3 8F    ; D0 ;
  5C E7 53 54 55 56 57 58 59 5A A0 85 8E E9 E4 D1    ; E0 ;
  30 31 32 33 34 35 36 37 38 39 B3 F7 F0 FA A7 FF    ; F0 ;
```

Syntax Rules for SBCS Translation Tables

- Blanks are used only as delimiters for readability purposes.
- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).

Customizing DBCS Translation Tables

Each DBCS translation table file contains more than one translation table. TCPHGLAT and TCPHGBIN, for example, contain EBCDIC to ASCII and ASCII to EBCDIC translation tables for both the KSC 5601 and Hangeul PC code pages.

The standard DBCS binary tables are used by the FTP server, SMTP server, and FTP client programs.

The figures on the following pages show examples of the standard source for the Kanji, Hangeul, and Traditional Chinese DBCS translation tables.

These source files contain two column pairs for each code page. The first column pair specifies double-byte EBCDIC to ASCII code point mappings for the indicated code page. The second column pair specifies double-byte ASCII to EBCDIC code point mappings for the indicated code page.

Existing code point mappings may be changed by simply overwriting the existing hexadecimal code. New code point mappings may be specified by adding a new

column pair with two double-byte hexadecimal codes. Code point mappings that are not specified, and are within the valid range for the code page, default to the "undefined" character, X'FFFF'.

The source file format allows EBCDIC to ASCII and ASCII to EBCDIC mappings to be specified separately. When adding or changing a code point mapping, care should be taken to modify both mappings for the code point. If, for example, a new mapping is added for EBCDIC to ASCII only, the ASCII to EBCDIC mapping for that code point will be the "undefined" character.

Any new code point mappings added outside the valid range for the corresponding code page will not be used by the programs that load the binary table.

DBCS Translation Table

The DBCS translation tables also contain SBCS code point mappings. These are used for mixed-mode DBCS strings, containing both SBCS and DBCS characters. Shift-out (X'0E') and shift-in (X'0F') characters are used on the EBCDIC host to denote the beginning and end of DBCS characters within a mixed-mode string.

The DBCS source files must contain exactly 256 SBCS code point mappings, situated at the end of the table. These may be modified to contain the required hexadecimal value.

Syntax Rules for DBCS Translation Tables

- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).
- Code point mappings in the file are position dependent. The first non-comment line for the DBCS and SBCS tables in the file will be used to establish the column position of the code point mappings, and must contain a conversion pair for each code page. Any conversion pairs on following lines must use the same column positions.
- It is permissible to leave blanks for code point mappings after the first line in the DBCS and SBCS areas. For example, if a line contains only one conversion pair, the column position will be used to determine which code page it refers to.
- The first column of each code page column pair, the "code index", must be in ascending numerical order. Any gaps in the ascending order will be marked as "undefined" in the binary table created by CONVXLAT.

Sample DBCS Translation Tables

The following examples are from the STANDARD DBCS translation table source files. Because these files are very large, only excerpts from the tables are shown. Ellipses (...) are used to indicate that information has been deleted.

Japanese Kanji DBCS Translation Tables

```
;
; STANDARD TCPKJLAT - Japanese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPKJBIN
; from this source file.
;
;DBCS Area - SJISETA,SJISATE
```

```
;          - JDECETA,JDECATE not used for STANDARD TCPKJBIN generation.
;
;SJISETA    SJISATE     JIS78ETA   JIS78ATE    JIS83ETA   JIS83ATE    ...
;
 4040 8140  8140 4040   4040 2121  2121 4040   4040 2121  2121 4040   ...
 4141 83BF  8141 4344   4141 2641  2122 4344   4141 2641  2122 4344   ...
 4142 83C0  8142 4341   4142 2642  2123 4341   4142 2642  2123 4341   ...
 4143 83C1  8143 426B   4143 2643  2124 426B   4143 2643  2124 426B   ...
 4144 83C2  8144 424B   4144 2644  2125 424B   4144 2644  2125 424B   ...
 4145 83C3  8145 4345   4145 2645  2126 4345   4145 2645  2126 4345   ...
 4146 83C4  8146 427A   4146 2646  2127 427A   4146 2646  2127 427A   ...
 4147 83C5  8147 425E   4147 2647  2128 425E   4147 2647  2128 425E   ...
 4148 83C6  8148 426F   4148 2648  2129 426F   4148 2648  2129 426F   ...
 4149 83C7  8149 425A   4149 2649  212A 425A   4149 2649  212A 425A   ...
:
:

;
; SBCS Area
;
;---------TCPKJBIN generation (no codefiles)--------------| |--------------
;SJISETA ATE   JIS78ETA ATE    JIS83ETA ATE   SJEUCETA ATE   J7KETA J7KATE
;
 00 00  00 00   00 00  00 00   00 00  00 00   00 00  00 00   00 00  00 00  ..
 01 01  01 01   01 01  01 01   01 01  01 01   01 01  01 01   01 01  01 01  ..
 02 02  02 02   02 02  02 02   02 02  02 02   02 02  02 02   02 02  02 02  ..
 03 03  03 03   03 03  03 03   03 03  03 03   03 03  03 03   03 03  03 03  ..
 04 1A  04 37   04 1A  04 37   04 1A  04 37   04 1A  04 37   04 1A  04 37  ..
 05 09  05 2D   05 09  05 2D   05 09  05 2D   05 09  05 2D   05 09  05 2D  ..
 06 1A  06 2E   06 1A  06 2E   06 1A  06 2E   06 1A  06 2E   06 1A  06 2E  ..
 07 7F  07 2F   07 7F  07 2F   07 7F  07 2F   07 7F  07 2F   07 7F  07 2F  ..
 08 1A  08 16   08 1A  08 16   08 1A  08 16   08 1A  08 16   08 1A  08 16  ..
 09 1A  09 05   09 1A  09 05   09 1A  09 05   09 1A  09 05   09 1A  09 05  ..
:
:
```

Hangeul DBCS Translation Tables

```
;
; STANDARD TCPHGLAT - Korean translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
;use CONVXLAT to generate STANDARD TCPHGBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID - 951      Code Page ID - 926
; KSCETA     KSCATE        HANETA     HANATE
;
  4040 A1A1  8FA1 D541     4040 8140  8140 4040
  4141 A1A2  8FA2 D542     4141 8141  8141 4141
  4142 A1A3  8FA3 D543     4142 8142  8142 4142
  4143 A1A4  8FA4 D544     4143 8143  8143 4143
  4144 A1A5  8FA5 D545     4144 8144  8144 4144
  4145 A1A6  8FA6 D546     4145 8145  8145 4145
  4146 A1A7  8FA7 D547     4146 8146  8146 4146
  4147 A1A8  8FA8 D548     4147 8147  8147 4147
  4148 A1A9  8FA9 D549     4148 8148  8148 4148
  4149 A1AA  8FAA D54A     4149 8149  8149 4149
:
:

;
; SBCS Area
;
;Code Page ID 1088   Code Page ID 891
;SKSCETA   SKSCATE   SHANETA   SHANATE
```

```
;
 00 00     00 00     00 00     00 00
 01 01     01 01     01 01     01 01
 02 02     02 02     02 02     02 02
 03 03     03 03     03 03     03 03
 04 FF     04 37     04 FF     04 37
 05 09     05 2D     05 09     05 2D
 06 FF     06 2E     06 FF     06 2E
 07 1C     07 2F     07 1C     07 2F
 08 FF     08 16     08 FF     08 16
 09 FF     09 05     09 FF     09 05
:
```

Traditional Chinese DBCS Translation Tables

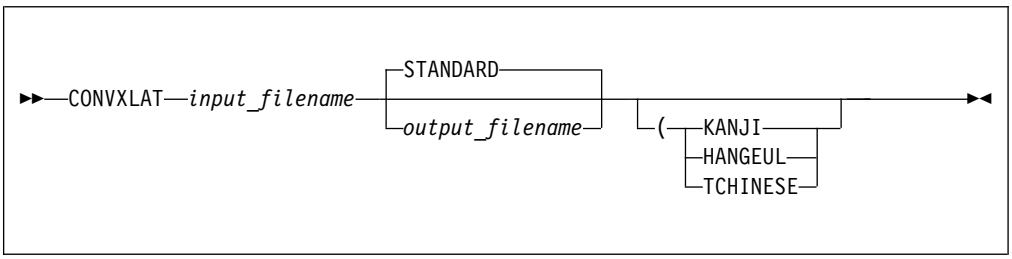
```
;
; STANDARD TCPCHLAT - Traditional Chinese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPCHBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID 927
; TCHETA      TCHATE
;
  4040 8140   8140 4040
  4141 83BF   8141 4344
  4142 83C0   8142 4341
  4143 83C1   8143 426B
  4144 83C2   8144 424B
  4145 83C3   8145 4345
  4146 83C4   8146 427A
  4147 83C5   8147 425E
  4148 83C6   8148 426F
  4149 83C7   8149 425A
:

;
; SBCS Area
;
; STCHETA  STCHATE
;
 00  00   00  00
 01  01   01  01
 02  02   02  02
 03  03   03  03
 04  cf   04  37
 05  09   05  2d
 06  d3   06  2e
 07  7f   07  2f
 08  d4   08  16
 09  d5   09  05
:
```

Converting Translation Tables to Binary

The CONVXLAT command converts a translation table source file to a binary file that usable by TCP/IP client and server programs. CONVXLAT may be used to convert both SBCS and DBCS source file.

CONVXLAT Command



Purpose

Used to convert a translation table source file to a binary file that is usable by TCP/IP client and server programs. May be used to convert both SBCS and DBCS source file.

Operands

input_filename
Specifies the file name of the source file to be converted. The source file must have a file type of TCPXLATE for SBCS tables, or a file type of TCPKJLAT, TCPHGLAT, or TCPCHLAT for DBCS tables. The first file with the required file name and file type in the standard minidisk search order is used.

output_filename
Specifies the destination file name created by the conversion. If this parameter is not specified, it defaults to the name STANDARD. The destination file type will be TCPXLBIN for SBCS tables, or TCPKJBIN, TCPHGBIN, or TCPCHBIN for DBCS tables. The destination file mode is A, which must be accessed in write mode.

KANJI
Specifies that the table being converted is the Kanji DBCS/SBCS translation table. The file type of the source file must be TCPKJLAT. The file type of the destination file will be TCPKJBIN.

HANGEUL
Specifies that the table being converted is the Hangeul DBCS/SBCS translation table. The file type of the source file must be TCPHGLAT. The file type of the destination file will be TCPHGBIN.

TCHINESE
Specifies that the table being converted is the Traditional Chinese DBCS/SBCS translation table. The file type of the source file must be TCPCHLAT. The file type of the destination file will be TCPCHBIN.

If no optional parameters are specified, then *input_filename* is assumed to contain an SBCS translation table.

Appendix A. Specifying Files and Data Sets

This appendix describes the file naming formats for the AIX, OS/390, DOS, OS/2 and Windows operating systems, as well as for the AS/400 operating system. Examples of each format are provided to show how the files appear to a TCP/IP user who is logged on to the different operating systems.

AIX Files

For the Advanced Interactive Executive (AIX) operating system, data is stored in files. Related files are stored in a directory.

The following is the format of an AIX file name.

Parameter
Description

directory_name
Specifies a directory name. Directories contain the names of files, other directories, or both.

file_name
Specifies a file name. It can be up to 14 characters long.

The complete name of an AIX file contains the directory name and the file name. The following is an example of an AIX file.

```
/mailfiles/cooks
```

Where:
mailfiles
Is the directory name.
cooks Is the file name.

In AIX, you specify the first slash (/) only when you begin at the root directory. If you are specifying a file in the current directory, enter only the file name. For example, if you are in the current directory *mailfiles* and you want to access the *cooks* file, specify:

```
cooks
```

The directory name and file name can each be up to 14 characters. The AIX operating system distinguishes between uppercase and lowercase letters in file names.

A directory name and file name should not include characters, such as backslash (\), ampersand (&), and period (.), which have a special meaning to the AIX operating system shell.

For more information about AIX files, see *AIX System User's Guide*.

© Copyright IBM Corp. 1987, 2016

OS/390 Data Sets

The two most common data set types used for storing user files on an OS/390 operating system are sequential data sets and partitioned data sets.

Sequential Data Sets

A sequential data set is a single file that can be allocated with any record length specified. The naming requirements for a sequential data set on an OS/390 host are minimal, and most of the requirements apply to any data set name under OS/390.

The naming requirements for a sequential data set are:

- No part of the name can start with a numeric.
- No part of the name can be more than eight characters in length.
- Each part of the name is separated by a period.
- If single quotation marks are not used when specifying the data set name, the OS/390 system appends the logon user ID as the first part of the name.

A sequential data set name can have a minimum of two and a maximum of 44 characters.

The following examples show the naming conventions for sequential data sets on an OS/390 host.

To access the sequential data set `KC00852.NAMES`, the user KC00852 enters one of the following:

```
'KC00852.NAMES'
     or
NAMES
```

Either of these formats is acceptable to access a sequential data set.

Partitioned Data Sets

A partitioned data set (PDS) is a group of files contained in a library. The individual files that make up a PDS are called members. You can access an entire PDS or any individual member of a PDS.

The following restrictions apply to the naming conventions that are used with a partitioned data set:

- No part of the name can start with a numeric.
- No part of the name can be more than eight characters in length.
- Each part of the name is separated by a period.
- If single quotation marks are not used when specifying the PDS name, the OS/390 system appends the logon user ID as the first part of the name.

The difference between a sequential and partitioned data set specification is that the partitioned data set user accesses the directory of members in the PDS, and the sequential data set user accesses an individual file.

The following examples show the naming conventions for partitioned data sets on an OS/390 host.

To access the partitioned data set `KC00852.PDS.NAMES`, the user KC00852 enters one of the following:

```
'KC00852.PDS.NAMES'
     or
PDS.NAMES
```

Either of these formats is acceptable to access a partitioned data set.

To access an individual file in a PDS, the member name is entered in parentheses.

To access the member PROPER in the PDS KC00852.PDS.NAMES, the user KC00852 enters one of the following:

```
'KC00852.NAMES(PROPER)'
     or
NAMES(PROPER)
```

Either of these formats is acceptable to access members in a partitioned data set.

DOS, OS/2, and Windows Files

DOS, OS/2, and Windows files are stored on high-capacity storage device, usually fixed disks or to a lesser extent, a diskette. Files, the smallest unit of organization on a hard drive are organized in directories and branching subdirectories. A storage device is identified by a drive letter.

The complete name of a DOS, OS/2, or Windows file contains the storage device identifier, directory name, file name, and extension. The following is an example of a complete name for a DOS, OS/2, or Windows file:

```
C:\WP\MAIL.LST
```

In this example, the storage device identifier is C: and the directory name is WP, which could be a group of word processing files. The file name is MAIL, and the file extension is .LST.

In DOS, OS/2, and Windows, the device identifier is a drive letter that is assigned by the file system, followed by a colon. The drive letter, such as A, B, or C, can represent either a physical device or a logical device. If a device identifier is not specified, the default is the device from which the system was booted or the current drive.

You assign the directory name, preceded by a backslash. A directory name is optional. If a directory name is not specified, the system searches the current directory.

Normally, you would also assign a file name. In DOS, file names can be eight characters long and have a three character extension. The file extension is a character string that consists of one to three characters, preceded by a period. A file extension is optional.

With OS/2, you will generally use one of two file systems: File Allocation Table or FAT, or else the High Performance File System, known as HPFS. You can have both these file systems active at the same time. For example, you can have the FAT file system on one hard disk, and the HPFS active on another.

The FAT format uses the familiar but limited naming convention, where a file or directory name can have up to eight letters, followed by a period and then the three-letter extension. HPFS provides support for long file names, up to 254 characters, including path information. including path information.

Windows supports 256-character filenames as well as upper and lowercase characters.

If you use an invalid file name, the system gives you an error message.

The following are examples of different DOS OS/2, and Windows file names that are valid.

```
TEMP
START.BAT
A:COMMAND.COM
C:SPF\SPFPC.HLP
C:REPORTS\ThisISALongFilename.TXT   (Windows)
```

AS/400 Operating System

For the AS/400 operating system, data is stored in files.

The following is the format of an AS/400 file.

Parameter
Description

library Is a library name. Libraries contain the names of programs, files, and commands.

file.member
Is the file name.

In AS/400, files can have one or more members. Each file can consist of data records, source programs, or database definitions.

The FTP subcommand PUT is used to copy a local file member into a file at the remote host. The following is an example:

```
PUT TCPLIBA/FILEA.MBRA TCPLIBB/FILEA.MBRA
```

In this example, the PUT subcommand copies the file member `MBRA` in file `FILEA` into library `TCPLIBA` at the local host to `MBRA` in `FILEA` in library `TCPLIBB` at the remote host. If the member already exists at the remote host, it is overwritten.

Appendix B. Using the NETRC DATA File

This appendix describes the NETRC DATA File and how it is used by:

- FTP Client
- REXEC
- OPENVM MOUNT CMS Command

NETRC-capable commands use fully-qualified host domain names when attempting to match a *command* specified host name with those in the NETRC DATA file. Thus, you may specify fully-qualified host domain names on the command in the NETRC DATA file. Otherwise, the command will construct a fully-qualified host domain name for you, by concatenating the host name you have provided with the domain origin specified on the DOMAINORIGIN statement of the TCPIP DATA configuration file.

You maintain the NETRC DATA file on your own minidisk or directory. The first NETRC DATA file found in the CMS search order will be used when the command is invoked. Since the NETRC DATA file contains logon passwords, ensure you restrict access to this file using security measures appropriate for your environment. Do not keep this file on a disk or directory to which other persons have access. A file mode number of 0 will not adequately protect this file.

The format for entries in the NETRC DATA file follows:

```
machine foreign_host login user_id password password
```

The IPv6 addresses are currently available for FTP service only.

Several sample NETRC DATA file entries follow:

```
machine oddjob login anonymou password anonymou
machine 123.45.67.89 login guest password guest
machine 123:5678::9abc login guest password guest
machine oddjob.specific.domain.com login cibulama password onion1
machine filmore.east.com login bluespwr password albertk
machine rocketman login gordon password flash
```

If the password contains leading or trailing blanks, or beings with a single quotation mark, it must be surrounded by single quotation marks and any imbedded single quotation mark must be doubled. For example:

```
machine bronzedragon  login alan password ride'em cowboy
machine bluedragon login karen password ' and they''re off '
machine cisko  login ben password watch the wormhole
```

Using The NETRC DATA File with FTP

The NETRC DATA file provides an alternative to responding to FTP prompts for logon information when you connect to a foreign host. When a user name and password for a specific host are defined in this file, FTP will use those values instead of prompting for this information.

When the FTP command, or its OPEN subcommand is issued, FTP searches the NETRC DATA file for the first valid match on the specified host. If found, that host's user name and password are used when a connection to that host is

© Copyright IBM Corp. 1987, 2016

attempted. If no match is found for the host in question, you are prompted for a user name and password, unless the NOPROMPT option has been specified.

Using The NETRC DATA File with REXEC

The NETRC DATA file provides an alternative for specifying the REXEC *user_id* and *password* parameters. If these parameters are defined within this file for a specific host, they can be omitted when you issue a REXEC command against that host.

REXEC searches the NETRC DATA file for the first valid match on the specified host. If found, that host's user name and password are used when a connection to that host is attempted. If no match is found for the host in question, you are prompted for a user name and password, unless the **-k** option has been specified.

The following example shows a REXEC command for which the required login *userid* and *password* values have been supplied by a NETRC DATA file entry. In this example the DIR C: command has been issued against the OS/2 host FILMORE:

```
rexec filmore dir c:

 The volume label in drive C is OS2.
 The Volume Serial Number is A6B0
11-30-93   9:56a     <DIR>           0  .
11-30-93   9:56a     <DIR>           0  ..
 1-17-94   4:36p       374           0  AUTOEXEC.BAT
 1-05-94   4:15p     <DIR>           0  BITMAPS
11-30-93  11:37a     <DIR>           0  CMLIB
 4-20-94   2:18p      2968          35  CONFIG.LAP
 5-01-95   4:31p      3281           0  CONFIG.SYS
 4-06-95  12:01p      3333           0  CONFIG.TCP
11-30-93  11:00a     <DIR>        1567  DESKTOP
 6-08-94  11:18a     <DIR>           0  FTPTEMP
11-30-93  11:21a     <DIR>        1607  IBMCOM
 5-01-95   8:22a      1016           0  IBMLVL.INI
 4-20-94   1:22p     <DIR>           0  LANLK
 6-08-94   1:01p     <DIR>           0  NFSTEST
11-30-93   9:56a     <DIR>        4049  OS2
11-30-93   9:56a     <DIR>           0  PSFONTS
 5-14-93  10:02p     40415           0  README
11-30-93  11:00a     <DIR>           0  SPOOL
 4-06-95  11:52a        80           0  STARTUP.BAK
 4-06-95  12:01p        80           0  STARTUP.CMD
 4-20-94   2:37p     <DIR>           0  TCPIP
 1-17-94   2:18p     <DIR>         329  UTILS
       22 file(s)      65759 bytes used
                     9502208 bytes free
Ready;

                                                RUNNING  GD3VM0
```

Using the NETRC DATA File with the OPEN MOUNT CMS Command

The NETRC DATA file provides an alternative for specifying the *username* and *password* parameters on the OPENVM MOUNT command. OPENVM MOUNT is a CMS commnd, documented in the *z/VM: OpenExtensions Commands Reference* that provides NFS client support for z/VM. This allows you to NFS-mount remote file systems in your Byte File System (BFS) hierarchy.

If the *username* and *password* are defined within the NETRC DATA file for a specific foreign host, you can omit them from an OPENVM MOUNT command issued for that host.

Appendix C. Mapping Values for the APL2 Character Set

GDDMXD/VM has default mapping values for the APL2 character set. However, if the GDXAPLCS MAP file exists, the default mapping values are overridden.

Each entry in the GDXAPLCS MAP file (alternative character set) contains the mapping for a particular physical key that corresponds to three characters. The characters correspond to the physical key by:

- Pressing the key alone
- Pressing the key and the Shift key simultaneously
- Pressing the key and the Alt key simultaneously

GDXAPLCS MAP file entries must contain the following seven single byte hexadecimal values entered as EBCDIC characters.

- Value 1 is the hexadecimal keycode for the physical key.
- Values 2, 4, and 6 identify whether the character is in the primary or alternative character set for the emulated 3179G. If the character is in the primary set, the value is 0; if the character is in the alternative set, the value is 8.
- Values 3, 5, and 7 specify the EBCDIC code of the character in the character set.

The combination of values 2 and 3 define the bytes that describe the character when the key corresponding to the keycode is pressed alone.

The combination of values 4 and 5 define the bytes that describe the character when the key corresponding to the keycode and the Shift key are pressed simultaneously.

The combination of values 6 and 7 define the bytes that describe the character when the key corresponding to the keycode and the Alt key are pressed simultaneously.

Table 41 lists the mapping values for the APL2 character set.

Table 41. Mapping Values for the APL2 Character Set

Character Name	Character Set Value	EBCDIC Value	Default Keycode
Quad Jot	8	73	9 + Shift
Quad Slope	8	CE	9 + Alt
1	0	F1	A
Diaeresis	8	72	A + Shift
Down Tack Up Tack	8	DA	A + Alt
2	0	F2	B
Overbar	8	A0	B + Shift
Del Tilde	8	FB	B + Alt
3	0	F3	C
<;	0	4C	C + Shift
Del Stile	8	DC	C + Alt
4	0	F4	D

© Copyright IBM Corp. 1987, 2016

Table 41. Mapping Values for the APL2 Character Set (continued)

Character Name	Character Set Value	EBCDIC Value	Default Keycode
Not Greater	8	8C	D + Shift
Delta Stile	8	DD	D + Alt
5	0	F5	E
=	0	7E	E + Shift
Circle Stile	8	CD	E + Alt
6	0	F6	F
Not Less	8	AE	F + Shift
Circle Slope	8	CF	F + Alt
7	0	F7	10
>;	0	6E	10 + Shift
Circle Bar	8	ED	10 + Alt
8	0	F8	11
Not Equal	8	BE	11 + Shift
Circle Star	8	FD	11 + Alt
9	0	F9	12
Down Caret	8	78	12 + Shift
Down Caret Tilde	8	CB	12 + Alt
0	0	F0	13
Up Caret	8	71	13 + Shift
Up Caret Tilde	8	CA	13 + Alt
+	0	4E	14
-	0	60	14 + Shift
!	8	DB	14 + Alt
Times	8	B6	15
Divide	8	B8	15 + Shift
Quad Divide	8	EE	15 + Alt
Q	0	D8	19
?	0	6F	19 + Shift
Q Underbar	8	58	19 + Alt
W	0	E6	1A
Omega	8	B4	1A + Shift
W Underbar	8	66	1A + Alt
E	0	C5	1B
Epsilon	8	B1	1B + Shift
E Underbar	8	45	1B + Alt
R	0	D9	1C
Rho	8	B3	1C + Shift
R Underbar	8	59	1C + Alt
T	0	E3	1D

Table 41. Mapping Values for the APL2 Character Set (continued)

Character Name	Character Set Value	EBCDIC Value	Default Keycode
Tilde	8	80	1D + Shift
T Underbar	8	63	1D + Alt
Y	0	E8	1E
Up Arrow	8	8A	1E + Shift
Y Underbar	8	68	1E + Alt
U	0	E4	1F
Down Arrow	8	8B	1F + Shift
U Underbar	8	64	1F + Alt
I	0	C9	20
Iota	8	B2	20 + Shift
I Underbar	8	49	20 + Alt
O	0	D6	21
Circle	8	9D	21 + Shift
O Underbar	8	56	21 + Alt
P	0	D7	22
Star	0	5C	22 + Shift
P Underbar	8	57	22 + Alt
Left Arrow	8	9F	23
Right Arrow	8	8F	23 + Shift
Quad Quote	8	DE	23 + Alt
Left Brk Right Brk	8	CC	24
Iota Underbar	8	74	24 + Shift
Delta Underbar	8	FC	24 + Alt
Equal Underbar	8	E1	25
Epsilon Underbar	8	E1	25 + Shift
Diaeresis Dot	8	75	25 + Alt
A	0	C1	27
Alpha	8	B0	27 + Shift
A Underbar	8	41	27 + Alt
S	0	E2	28
Up Stile	8	8D	28 + Shift
S Underbar	8	62	28 + Alt
D	0	C4	29
Down Stile	8	8E	29 + Shift
D Underbar	8	44	29 + Alt
F	0	C6	2A
Underbar	0	6D	2A + Shift
F Underbar	8	46	2A + Alt
G	0	C7	2B

Table 41. Mapping Values for the APL2 Character Set (continued)

Character Name	Character Set Value	EBCDIC Value	Default Keycode
Del	8	BA	2B + Shift
G Underbar	8	47	2B + Alt
H	0	C8	2C
Delta	8	BB	2C + Shift
H Underbar	8	48	2C + Alt
J	0	D1	2D
Jot	8	AF	2D + Shift
J Underbar	8	51	2D + Alt
K	0	D2	2E
Quote	0	7D	2E + Shift
K Underbar	8	52	2E + Alt
L	0	D3	2F
Quad	8	90	2F + Shift
L Underbar	8	53	2F + Alt
Left Bracket	8	AD	30
(	0	4D	30 + Shift
Down Tack Jot	8	FE	30 + Alt
Right Bracket	8	BD	31
)	0	5D	31 + Shift
Up Tack Jot	8	EF	31 + Alt
Z	0	E9	36
Left Shoe	8	9B	36 + Shift
Z Underbar	8	69	36 + Alt
X	0	E7	37
Right Shoe	8	9A	37 + Shift
X Underbar	8	67	37 + Alt
C	0	C3®	38
Up Shoe	8	AA	38 + Shift
C Underbar	8	43	38 + Alt
V	0	E5	39
Down Shoe	8	AB	39 + Shift
V Underbar	8	65	39 + Alt
B	0	C2	3A
Down Tack	8	AC	3A + Shift
B Underbar	8	42	3A + Alt
N	0	D5	3B
Up Tack	8	BC	3B + Shift
N Underbar	8	55	3B + Alt
M	0	D4	3C

Table 41. Mapping Values for the APL2 Character Set (continued)

Character Name	Character Set Value	EBCDIC Value	Default Keycode
Stile	0	4F	3C + Shift
M Underbar	8	54	3C + Alt
,	0	6B	3D
;	0	5E	3D + Shift
Up Shoe Jot	8	DF	3D + Alt
period	0	4B	3E
:	0	7A	3E + Shift
Slope Bar	8	EB	3E + Alt
/	0	61	3F
\	0	E0	3F + Shift
Slash Bar	8	EA	3F + Alt
Space	0	40	45

Appendix D. Using DBCS with FTP and Mail

This appendix describes how to use the DBCS facilities provided by FTP and SMTP.

Using DBCS with FTP

This section describes how to use FTP with DBCS support to exchange DBCS files between hosts supporting DBCS file transfer.

DBCS Translation Tables

The VM TCP/IP FTP server and client programs access files containing data that is usually in EBCDIC format. To transfer these files to or from an ASCII based host requires the use of a translation table.

The FTP server and client may be configured to load a number of DBCS translation tables. These are used during file transfers to convert EBCDIC DBCS characters to and from ASCII. The LOADDBCSTABLE statement in FTP DATA is used by both the FTP server and client to determine which DBCS translate table files should be loaded at initialization time.

Control and Data Connection Translation

The transfer of a DBCS file by FTP actually uses three different translation tables; two SBCS and one DBCS. The control connection that is used to transfer FTP commands and replies, uses the SBCS translation table that is normally loaded from the STANDARD TCPXLBIN file. The data connection that is used to transfer the file, uses both an SBCS and DBCS translation table that are normally loaded from either the STANDARD TCPKJBIN, STANDARD TCPHGBIN, or STANDARD TCPCHBIN file.

Both SBCS and DBCS translation tables are required for the transfer of a DBCS file, as the file may contain mixed-mode strings. A mixed mode string contains both SBCS and DBCS data, delimited by shift-out and shift-in characters.

Although DBCS support for FTP does not include direct support for Katakana, it is possible to separately configure the SBCS table used for the control connection (user interface), and the data connection. The SBCS translation table for SJISKANJI, for example, could be altered to a Katakana based code page.

DBCS Subcommands

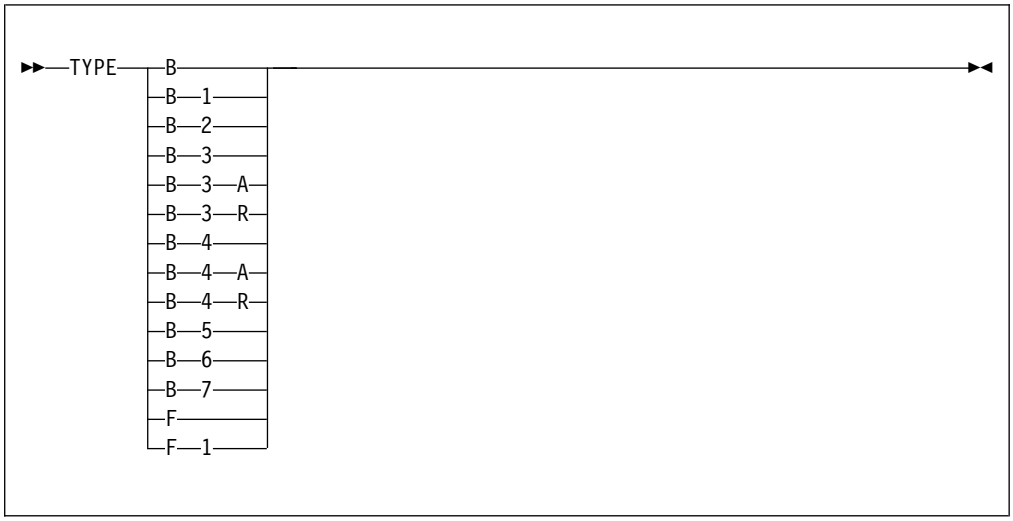
DBCS files are transferred using the standard FTP subcommands “put” and “get”. Before the transfer commences, however, the current transfer type for the session must be set to the required DBCS type. To set the transfer type to DBCS for an FTP session, requires the sending of a TYPE command from the client to the server in the correct format. The VM TCP/IP FTP client provides three ways of generating TYPE commands:

1. TYPE subcommand
2. TYPE subcommand aliases
3. QUOTE subcommand

© Copyright IBM Corp. 1987, 2016

TYPE Subcommand

The TYPE command sets the transfer type for the client and server at the same time with one command. The following DBCS TYPE subcommands are supported by the VM TCP/IP FTP server and client.



Parameter
Description

TYPE B
Change current transfer type to Shift JIS Kanji

TYPE B 1
Change current transfer type to Shift JIS Kanji

TYPE B 2
Change current transfer type to Extended Unix Code Kanji

TYPE B 3
Change current transfer type to JIS 1983 Kanji using ASCII shift-in escape sequence `ESC ( B`

TYPE B 3 A
Change current transfer type to JIS 1983 Kanji using ASCII shift-in escape sequence `ESC ( B`

TYPE B 3 R
Change current transfer type to JIS 1983 Kanji using JISROMAN shift-in escape sequence `ESC ( J`

TYPE B 4
Change current transfer type to JIS 1978 Kanji using ASCII shift-in escape sequence `ESC ( B`

TYPE B 4 A
Change current transfer type to JIS 1978 Kanji using ASCII shift-in escape sequence `ESC ( B`

TYPE B 4 R
Change current transfer type to JIS 1978 Kanji using JISROMAN shift-in escape sequence `ESC ( J`

TYPE B 5
Change current transfer type to Hangeul

TYPE B 6
Change current transfer type to Korean Standard Code KSC-5601, 1989 version

TYPE B 7
Change current transfer type to Traditional Chinese (5550)

TYPE F
Change current transfer type to IBM (EBCDIC) Kanji

TYPE F 1
Change current transfer type to IBM (EBCDIC) Kanji

TYPE Subcommand Aliases

Each DBCS TYPE subcommand may be specified using a single word alias. Each TYPE subcommand alias generates the same TYPE command to the client and server as the corresponding TYPE subcommand. The TYPE subcommand aliases may be abbreviated using the minimum unambiguous client subcommand abbreviations.

The TYPE subcommand aliases also have an optional parameter (NOTYPE, that specifies that the DBCS type should only be set locally. This is used when connecting to an FTP server that does not support the TYPE command generated by the TYPE subcommand alias. For example, SJISKANJI (NOTYPE causes the FTP client to change its transfer type to Shift JIS Kanji, without changing the transfer type in the FTP server. The server in this example should be set to the ASCII transfer type before the (NOTYPE subcommand is issued. Table 42 indicates the actual server command that would be generated for each client subcommand alias:

Table 42. FTP TYPE commands

Client Subcommand	Server Command	Description
SJISKANJI	TYPE B 1	Shift JIS Kanji transfer type
EUCKANJI	TYPE B 2	Extended Unix Code Kanji transfer type
JIS83KJ	TYPE B 3	JIS 1983 Kanji using ASCII shift-in transfer type
JIS83KJ (ASCII	TYPE B 3 A	JIS 1983 Kanji using ASCII shift-in transfer type
JIS83KJ (JISROMAN	TYPE B 3 R	JIS 1983 Kanji using JISROMAN shift-in transfer type
JIS78KJ	TYPE B 4	JIS 1978 Kanji using ASCII shift-in transfer type
JIS78KJ (ASCII	TYPE B 4 A	JIS 1978 Kanji using ASCII shift-in transfer type
JIS78KJ (JISROMAN	TYPE B 4 R	JIS 1978 Kanji using JISROMAN shift-in transfer type
HANGEUL	TYPE B 5	Hangeul transfer type
KSC5601	TYPE B 6	Korean Standard Code KSC-5601 transfer type
TCHINESE	TYPE B 7	Traditional Chinese (5550) transfer type
IBMKANJI	TYPE F 1	IBM (EBCDIC) Kanji transfer type

QUOTE Subcommand

The QUOTE subcommand may be used to send an uninterpreted string of data to the server. The QUOTE subcommand may be used to generate any of the DBCS TYPE commands supported by the server. The QUOTE subcommand would be used when the FTP server supports the DBCS TYPE command, but the FTP client does not. For example, QUOTE TYPE B 1 causes the FTP server to change its

transfer type to Shift JIS Kanji, without changing the transfer type in the FTP client. The client in this example should be set to the ASCII transfer type before the QUOTE subcommand is issued.

The following examples show the screen display when setting the DBCS transfer type to JIS78KJ, shift-in JISROMAN, using a VM TCP/IP FTP client connected to a VM TCP/IP FTP server. All three methods of setting the DBCS transfer type are demonstrated.

```
User:      jis78kj (jisroman
System:    >;>;>;TYPE b 4 r
System:    200 Representation type is KANJI JIS 1978 shift-in JISROMAN
System:    Command:
User:      type b 4 r
System:    >;>;>;TYPE b 4 r
System:    200 Representation type is KANJI JIS 1978 shift-in JISROMAN
System:    Command:
User:      jis78kj (jisroman notype
System:    Command:
User:      quote type b 4 r
System:    >;>;>;type b 4 r
System:    200 Representation type is KANJI JIS 1978 shift-in JISROMAN
System:    Command:
```

Defining FTP with Kanji Support

The following FTP subcommands can set the appropriate transfer type for the Kanji file transfer. The translation table STANDARD TCPKJBIN converts the transmitted data between Kanji and ASCII. This file is installed in the user visible minidisk from FTPSERVE user ID. The following are the FTP Kanji transfer type subcommands:

Subcommand
Option

SJISKANJI
(notype

EUCKANJI
(notype

JIS83KJ
(notype

JIS78KJ
(notype

IBMKANJI
(notype

The (notype option is used for the standard FTP server that does not have Kanji support. FTP sets the data type for the client and server at the same time with one command. If Kanji support is only in the FTP user, the FTP server cannot accept Kanji data types with the TYPE subcommand. The Kanji subcommands with the (notype option should be used, without the TYPE subcommand.

Using FTP with Kanji Support

The FTP TYPE subcommand supports single byte transfer for ASCII and EBCDIC data transfer (TYPE A or TYPE E). Kanji data types allow you to transfer text files with a Kanji code set. TYPE B defines the data type of the Kanji code based on the ASCII code set. TYPE F defines the data type of the Kanji code based on the EBCDIC code set.

The following are the Kanji code set types, based on ASCII (TYPE B) AND EBCDIC (TYPE F):

Command
Kanji Code Set
TYPE B 1
Shift JIS
TYPE B 2
Extended UNIX Code
TYPE B
JIS 1983 edition
TYPE B 4
JIS 1978 edition
TYPE F 1
IBM Kanji

The number associated with the data type is the argument of the TYPE command. For example, if JIS 1983 edition code is used in the data transfer, issue the command TYPE B 3. If IBM code is used in the data transfer, issue the command TYPE F 1.

If the FTP user does not support the Kanji extension, you should issue the FTP QUOTE subcommand to change the data type in the FTP Kanji server. For example, QUOTE TYPE B 1 causes the FTP server to change its data type to Shift JIS code without changing the data type in the FTP user. In this example, you must set the ASCII data type before you issue the QUOTE subcommand.

Using DBCS with Mail

This section describes how to use SMTP with DBCS support to exchange DBCS mail and messages between hosts supporting DBCS mail.

SMTP Server DBCS Support

Mail in EBCDIC DBCS Kanji, Hangeul, or Traditional Chinese, may be sent to a remote host via the CMS NOTE and SENDFILE commands, if the local SMTP server or agent is configured with the corresponding DBCS support. For more information on configuring DBCS support for the SMTP server, see *z/VM: TCP/IP Planning and Customization*.

SMTP Protocol for 8-bit characters

The protocol specification for SMTP describes a transport service that provides an 8-bit byte transmission channel, with each 7-bit character transmitted right-justified in an octet with the high-order bit cleared to zero. JIS Kanji DBCS may be transmitted using this protocol, as its code consists of two bytes with 7-bit ASCII and three bytes of escape sequences. JUNET, which is the Japanese academic and research network connecting various UNIX operating systems, provides network services using JIS Kanji code. Other DBCS types, such as Shift-JIS Kanji, require transmission using 8-bit characters. To allow transmission of these other types, the protocol has been extended in the VM SMTP server to accept 8-bit characters.

To successfully distribute mail with 8-bit DBCS characters, requires that other SMTP servers on the same network support this extension to the SMTP protocol. The AIX SMTP server and the MVS version 3.1 SMTP server also support the transmission of 8-bit characters.

Conversion of DBCS Mail

The transmission of DBCS mail by SMTP actually uses three different translation tables; two SBCS and one DBCS. Mail headers are converted to ASCII using the SBCS translation table that is normally loaded from the STANDARD TCPXLBIN file. The body of the mail is converted using both the SBCS and DBCS translation tables that are normally loaded from either the STANDARD TCPKJBIN, STANDARD TCPHGBIN, or STANDARD TCPCHBIN file.

Both SBCS and DBCS translation tables are required for the transmission of DBCS mail, as the mail may contain mixed-mode strings. A mixed mode string contains both SBCS and DBCS data, delimited by shift-out and shift-in characters.

Although DBCS support for SMTP does not include direct support for Katakana, it is possible to separately configure the SBCS table used for mail headers, and that used for the body of the mail. If the SMTP server was configured with SJISKANJI, for example, then the SBCS translation table for SJISKANJI could be altered to a Katakana based code page.

DBCS conversion is only performed on outgoing and incoming mail to and from other hosts. Mail spooled to SMTP via NOTE or SENDFILE for the local host, is delivered directly, without any DBCS code conversion.

Conversion of DBCS Files

The transmission of DBCS files by UFT uses three different translation tables; two SBCS and on DBCS. UFT protocol statements are converted to ASCII using the SBCS translation table imbedded in the UFT client (which maps to the STANDARD table). The file data is converted using both the SBCS and DBCS translation tables that are normally loaded from the filename specified on the SENDFILE TRANSLATE option with a filetype of either TCPKJBIN, TCPHGBIN or TCPCHBIN.

Appendix E. Management Information Base Objects

This appendix describes the objects defined by the Management Information Base (MIB) and MIB-IIvariables supported by VM.

The following list contains the supported variables.
- System
- Interfaces
- Address Translation
- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Bridge (dot1dBridge)

For a complete list of the MIB and MIB-II variables, see RFC 1156 (MIB) and RFC 1158 (MIB-II).

The object types are defined using the following fields:

Object
A textual name, called the OBJECT DESCRIPTOR, for the object type, along with its corresponding OBJECT IDENTIFIER.

Syntax
The syntax for the object type, using ASN.1 notation. The syntax indicates that the following data is to be treated as a collection of objects that can be repeated. For example, the collection of objects can be row entries in a routing table with an unspecified number of rows. Therefore, these descriptors are not really MIB objects that can be manipulated, but only the objects within the sequence.

Definition
A description of the object type. Because this MIB is intended for use in multivendor environments, object types must have consistent meanings across all machines.

Access
One of read-only, read-write, write-only, or not-accessible.

VM Support
One of yes, no, or read-only.

Structure and Identification of Management Information (SMI)

Managed objects are accessed through a virtual information store, known as the Management Information Base (MIB). Objects in the MIB are defined using Abstract Syntax Notation One (ASN.1).

Each object type has a name, a syntax, and an encoding description. The name is represented uniquely as an object identifier, which assigned by a systems administrater.

The syntax for an object type defines the abstract data structure corresponding to that object type. For example, the structure of a given object type might be an

© Copyright IBM Corp. 1987, 2016

integer or an octet string. RFC 1155 restricts the ASN.1 constructs that can be used. These restrictions are made solely for the sake of simplicity.

The encoding of an object type describes how instances of that object type are represented using the object's type syntax. RFC 1155 specifies the use of the basic encoding rules of ASN.1.

Names

Names are used to identify managed objects and they are hierarchical in nature. For example, each international standard has an object identifier assigned to it for the purpose of identification. In short, object identifiers are a means for identifying some object, regardless of the semantics associated with it.

The root node itself is unlabeled, but has at least three nodes directly under it: one node is administered by the International Standards Organization (ISO), with label iso(1); another is administered by the International Telegraph and Telephone Consultative Committee (CCITT), with label ccitt(2); and the third is jointly administered by the ISO and the CCITT, joint-iso-ccitt(3).

Under the iso(1) node, the ISO has designated one subtree for use by other (inter)national organizations, org(3). Under this node, one of the subtrees has been allocated to the US Department of Defense (DOD), dod(6), under which only one node has been assigned to the Internet community, and it is administered by the Internet Activities Board (IAB). Therefore, the Internet subtree of object identifiers starts with the prefix 1.3.6.1.

The Management Subtree

Under the IAB node, four subtrees have been defined, namely, directory(1), mgmt(2), experimental(3), and private(4). The administration of the mgmt(2) subtree was delegated by the IAB to the Internet Assigned Numbers Authority for the Internet. This subtree is used to identify objects that are defined in IAB-approved documents. So far, the prefix of the object identifiers is: 1.3.6.1.2.

When RFCs, which define new versions of the Internet-standard MIB, are approved, they are assigned an object identifier for identifying the objects defined by that RFC. The Internet-standard MIB has been assigned management document number one. Therefore, its object identifier is:

```
   { mgmt 1 }
or
   1.3.6.1.2.1
```

The private(4) subtree is used to define enterprise-specific variables in the MIB, which means that you can add your own objects to the MIB. The prefix of these variables is: 1.3.6.1.4.

Figure 63 on page 469 illustrates the hierarchical ISO tree that has been adopted to identify managed objects.

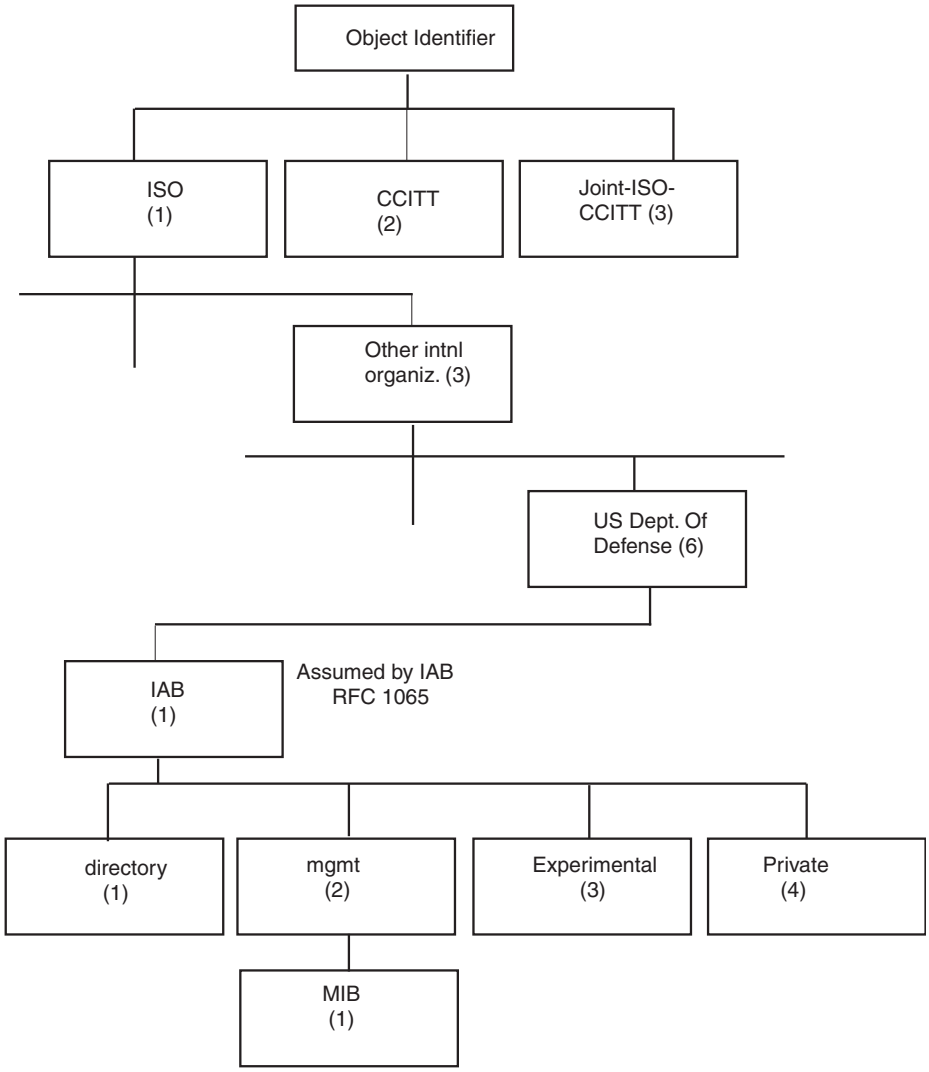


Figure 63. The SMI Hierarchical Tree

The SMI is not supposed to define objects in the MIB, but it does specify a format to be used by the RFCs that define these objects. An object type definition consists of the following fields.

Field Name
Description

Object
Specifies a textual name, termed the object descriptor, for the object type, along with its corresponding object identifier

Syntax
Specifies the abstract syntax for the object type, such as integer, octet string, counter, timeticks, and so on

Definition
Specifies a textual description of the semantics of the object type

Access
Specifies read-only, read-write, write-only or not-accessible

Status Specifies mandatory, optional, or obsolete

The following is an example of an object type definition:

Object:
sysDescr.
Syntax:
Octet string.
Definition:
A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. It is mandatory that this only contain printable ASCII characters.
Access:
read-only.
Status:
mandatory.

For more information, see RFC 1155.

MIB/Network Elements

The MIB defines the information that can be obtained from SNMP agents. The MIB defines objects, such as packet counts and routing tables that are relevant to a TCP/IP environment. The objects defined by the MIB are divided into groups, with each group representing a set of management data.

Currently, the following groups have been defined:

Group Name
Description

System
Contains information about the entity, such as system hardware, software, and the version number.

Interfaces
Contains all the interfaces through which the nodes can send and receive IP datagrams. It also contains counters for packets sent and received and errors.

Address translation
Contains information for mapping a network address into a specific subnetwork or physical address. This group is deprecated, meaning that it still exists but will be deleted in the future.

IP Contains information about the IP layer, such as the number of datagrams sent, received, and forwarded. It includes two tables: the IP address table contains the IP addressing information for the entity; the IP routing table contains one entry for each route presently known to it.

ICMP Contains the ICMP input and output statistics.

TCP Contains information about the TCP connections, such as the maximum number of connections the entity can support, the total number of retransmitted segments, the minimum and maximum time-out values, and so on.

UDP Contains information about the UDP layer, such as counters for datagrams sent and received.

EGP Contains information about EGP peers, such as the number of messages sent and received, error counters, and so on.

Transmission
Contains media-specific information. This group is not currently implemented.

SNMP
Contains information about the SNMP agent, such as the number of SNMP packets received, the number of SNMP requests with bad community names, and so on.

Bridge
Contains information about bridges and devices used to connect Local Area Network segments below the network layer. The VSWITCH is z/VM's implementation of a virtual bridge.

The system group is the first group in the MIB. Therefore, it is identified as:

```
{ mib 1 }
```
or
```
1.3.6.1.2.1.1
```

Each group has its own subtree. For example, the first variable in the system group is the system description (sysDescr), so that it can be represented as:

```
{ system 1 }
```
or
```
1.3.6.1.2.1.1.1
```

To summarize the explanation, the following describes the composition of the ASN.1 object identifier for sysDescr.

```
iso  org  dod  internet  mgmt  mib  system  sysDescr
 1    3    6       1       2     1      1        1
```

System Group

Table 43 lists the objects in the system group. The system objects identify the type of system with a text description and the vendor-assigned object-ID as an identification to the type of SNMP server.

Table 43. Implementation of the System Group

Object	Syntax	Definition	Access
sysDescr { system 1 }	DisplayString	A description of the entry. This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. This description must only contain printable ASCII characters.	read-only

1. All accesses of read-write indicate a read-only access for VM.

Table 43. Implementation of the System Group (continued)

Object	Syntax	Definition	Access
sysObjectID { system 2 }	OBJECT IDENTIFIER	The vendor's authorization identification of the network management subsystem contained in the entry. This value is allocated within the Structure for Management Information (SMI) enterprise's subtree (1.3.6.1.4.1) and provides an easy and clear means for determining *what kind of box* is being managed. For example, if vendor *Stones, Inc.* was assigned the subtree 1.3.6.1.4.1.42, it could assign the identifier 1.3.6.1.4.1.42.1.1 to the router *Fred Router*.	read-only
sysUpTime { system 3 }	TimeTicks	The time (in hundredths of a second) since the network management portion of the system was last started. The maximum value of the time is around 248 days. When the value of the time is greater than the maximum value, the counter will wrap back to zero.	read-only
sysContact { system 4 }	DisplayString	The textual identification of the contact person for this managed node, together with information about how to contact this person.	read-write[1]
sysName { system 5 }	DisplayString	An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name	read-write[1]
sysLocation { system 6 }	DisplayString	The physical location of this node (for example, *telephone closet, 3rd floor*)	read-only

Table 43. Implementation of the System Group (continued)

Object	Syntax	Definition	Access
sysServices { system 7 }	INTEGER	A value that indicates the set of services that this entity potentially offers. The value is a sum. This sum initially takes the value 0, then, for each layer L in the range 1 through 7 for which this node performs transactions, 2 raised to (L - 1) is added to the sum. For example, a node that performs only routing functions would have a value of 4 (2^(3-1)). In contrast, a node that is a host offering application services would have a value of 72 (2^(4-1) + 2^ (7-1)). Note that in the context of the internet suite of protocols, values should be calculated accordingly: Layer Functionality **1** Physical (for example, repeaters) **2** Datalink/subnetwork (for example, bridges) **3** Internet (for example, supports the IP) **4** End-to-end (for example, supports the TCP) **7** Applications (for example, supports the SMTP) For systems including OSI protocols, layers 5 and 6 can also be counted.	read-only

Interfaces Group

Table 44lists the objects in the interfaces group. The interfaces objects are a set of entries for each network interface below the IP layer that can send and receive datagrams.

Table 44. Implementation of the Interfaces Group

Object	Syntax	Definition	Access
ifNumber { interfaces 1 }	INTEGER	The number of network interfaces (regardless of their current state) present on this system.	read-only
ifTable { interfaces 2 }	SEQUENCE of IfEntry	A list of interface entries. The number of entries is given by the value of ifNumber.	not applicable

Table 44. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifEntry { ifTable 1 }	IfEntry ::= SEQUENCE ifIndex INTEGER, ifDescr (DISPLAY STRING in MIB-II) ifType INTEGER, ifMtu INTEGER, ifSpeed Gauge, ifPhysAddress OCTET STRING, ifAdminStatus INTEGER, ifOperStatus INTEGER, ifLastChange TimeTicks, ifInOctets Counter, ifInUcastPkts Counter, ifInNUcastPkts Counter, ifInDiscards Counter, ifInErrors Counter, ifInUnkownProtos Counter, ifOutOctets Counter, ifOutUcastPkts Counter, ifOutNUcastPkts Counter, ifOutDiscards Counter, ifOutErrors Counter, ifOutQLen Gauge ifSpecific Object ID	An interface entry that contains objects at the subnetwork layer and below for a particular interface.	read-write[1]
ifIndex { ifEntry 1 }	INTEGER	A unique value for each interface. Values range between 1 and the value of ifNumber. The value for each interface must remain constant for at least one start of the systems network management system to the next start.	read-only

Table 44. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifDescr { ifEntry 2 }	DisplayString	A text string containing information about the interface. This string should include the name of the manufacturer, the product name, and the version of the hardware interface.	read-only
ifType { ifEntry 3 }	INTEGER other (1), regular 1822 (2), hdh1822 (3), ethernet-csmacd (6), iso88023-csmacd (7), iso88024-tokenBus (8), iso88025-tokenRing (9), iso88026-kman (10), starLan (11), proteon-10Mbit (12), proteon-80Mbit (13), hyperchannel (14), fddi (15), lapb (16), sdlc (17), tl-carrier (18), cept (19), basicISDN (20), primaryISDN (21), propPointToPointSerial (22), terminalServer-asypcPort (23), softwareLoopback (24), eon (25), ethernet-3Mbit (26), nsip (27), slip (28)	The type of interface, distinguished according to the physical/link/ network protocol(s) immediately *below* IP in the protocol stack.	read-only
ifMtu { ifEntry 4 }	INTEGER	The size of the largest datagram that can be sent or received on the interface, specified in octets. For interfaces that are used for transmitting IP datagrams, this is the size of the largest IP datagram that can be sent on the interface.	read-only
ifSpeed { ifEntry 5 }	Gauge	An estimate of the interface's current bandwidth in bits per second. For interfaces that do not vary in bandwidth or for those where no accurate estimate can be made, this object should contain the nominal bandwidth.	read-only
ifPhysAddress { ifEntry 6 }	OCTET STRING	The interface's address at the protocol layer immediately *below* IP in the protocol stack. For interfaces that do not have such an address (for example, a serial line), this object should contain an octet string of length 0.	read-only

Table 44. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifAdminStatus { ifEntry 7 }	INTEGER up (1), down (2), testing (3)	The desired state of the interface. The testing (3) state indicates that operational packets cannot be passed.	read-write[1]
ifOperStatus { ifEntry 8 }	INTEGER up (1), down (2), testing (3)	The current operational state of the interface. The testing (3) state indicates that operational packets cannot be passed.	read-only
ifLastChange { ifEntry 9 }	TimeTicks	The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered before the last start-up of the local network management subsystem, then this object contains a value of 0. The maximum value of the time is around 248 days. When the value of the time is greater than the maximum value, the counter will wrap back to zero.	read-only
ifInOctets { ifEntry 10 }	Counter	The total number of octets received on the interface, including framing characters.	read-only
ifInUcastPkts { ifEntry 11 }	Counter	The number of subnetwork-unicast packets delivered to a higher-layer protocol.	read-only
IfInNUcastPkts { ifEntry 12 }	Counter	The number of non-unicast (for example, subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.	read-only
ifInDiscards { ifEntry 13 }	Counter	The number of inbound packets that were chosen to be discarded even though errors had not been detected to prevent their delivery to a higher-layer protocol. One possible reason for discarding such a packet could be to free buffer space.	read-only
ifInErrors { ifEntry 14 }	Counter	The number of inbound packets that contain errors that prevent delivery to a higher-layer protocol.	read-only
ifInUnknownProtos { ifEntry 15 }	Counter	The number of packets received through the interface that were discarded because of an unknown or unsupported protocol.	read-only
ifOutOctets { ifEntry 16 }	Counter	The total number of octets transmitted out of the interface, including framing characters.	read-only
ifOutUcastPkts { ifEntry 17 }	Counter	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.	read-only

Table 44. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifOutNUcastPkts { ifEntry 18 }	Counter	The total number of packets that higher-level protocols request to be transmitted to a non-unicast (for example, a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.	read-only
ifOutDiscards { ifEntry 19 }	Counter	The number of outbound packets that were chosen to be discarded even though errors had not been detected to prevent their being transmitted. One reason for discarding such a packet could be to free buffer space.	read-only
ifOutErrors { ifEntry 20 }	Counter	The number of outbound packets that could not be transmitted because of errors.	read-only
ifOutQLen { ifEntry 21 }	Gauge	The length of the output packet queue (in packets).	read-only

ifSpecific
 	
{ ifEntry 22 }
	OBJECT IDENTIFIER	A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an ethernet, then the value of this object refers to a document defining objects specific to Ethernet. If an agent is not configured to have a value for any of these variables, the following object identifier is returned:
		`nullSpecific OBJECT IDENTIFIER ...`
		`::= { 0 0 }`
		
		Note that nullSpecific is a syntactically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.
			read-only

Address Translation Group

Table 45 lists the objects in the address translation group. The address translation objects are a set of entries for each network interface, below the IP layer, that can send and receive datagrams.

Table 45. Implementation of the Address Translation Group

Object	Syntax	Definition	Access
atTable { at 1 }	SEQUENCE OF AtEntry	The Address Translation tables contain the NetworkAddress to physical address equivalences. Some interfaces do not use translation tables to determine address equivalences. If all interfaces are of this type, then the Address Translation table is empty; it has 0 entries.	read-write[1]
atEntry { atTable 1 }	AtEntry ::= SEQUENCE atIfIndex INTEGER, atPhysAddress OCTET STRING, atNetAddress NetworkAddress	Each entry contains one NetworkAddress to the physical address equivalent.	read-write
atIfIndex { atEntry 1 }	INTEGER	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface that is identified by the same value of ifIndex.	read-write
atPhysAddress { atEntry 2 }	OCTET STRING	The media-dependent physical address.	read-write
atNetAddress { atEntry 3 }	NetworkAddress	The NetworkAddress (for example, the IP address) corresponding to the media-dependent physical address.	read-write

IP Group

Table 46 lists the objects in the IP group. The IP objects are the statistics and gateway routing tables for the IP layer.

Table 46. Implementation of the IP Group

Object	Syntax	Definition	Access
ipForwarding { ip 1 }	INTEGER gateway (1), — entry forwards datagrams host (2) — entry does NOT forward datagrams	Indicates if this entry is acting as an IP gateway for the forwarding of datagrams received by, but not addressed to, this entry. IP gateways forward datagrams; hosts do not, except those source-routed through the host.	read-only
ipDefaultTTL { ip 2 }	INTEGER	When a TTL value is not supplied by the transport layer protocol, the default value inserts into the time-to-live field of the IP header of datagrams that originate at this entry.	read-write[1]

Table 46. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipInReceives { ip 3 }	Counter	The number of input datagrams received from interfaces, including those received in error.	read-only
ipInHdrErrors { ip 4 }	Counter	The number of input datagrams discarded because of errors in their IP headers. For example, bad checksums, mismatched version number, format errors, time-to-live exceeded, and processing errors in IP options.	read-only
ipInAddrErrors { ip 5 }	Counter	The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entry. This count includes invalid addresses (for example, 0.0.0.0), addresses of unsupported classes (for example, Class E), and destination addresses that were not local addresses (for example, IP gateways).	read-only
ipForwDatagrams { ip 6 }	Counter	The number of input datagrams for which this entry is not their final IP destination. As a result, an attempt is made to find a route to their final destination. For entries that do not act as IP gateways, this count includes only those packets that are source-routed successfully through this entry.	read-only
ipInUnkownProtos { ip 7 }	Counter	The number of locally-addressed datagrams received successfully, but discarded because of an unknown or unsupported protocol.	read-only
ipInDiscards { ip 8 }	Counter	The number of input IP datagrams that are processed without problems, but are discarded (for example, for lack of buffer space). This count does not include any datagrams discarded while awaiting reassembly.	read-only
ipInDelivers { ip 9 }	Counter	The number of input datagrams successfully delivered to IP user-protocols including ICMP.	read-only
ipOutRequests { ip 10 }	Counter	The number of IP datagrams that are supplied to IP and ICMP in requests for transmission. This count does not include datagrams counted in ipForwDatagrams.	read-only
ipOutDiscards { ip 11 }	Counter	The number of output IP datagrams that transmit without problems, but are discarded (for example, for lack of buffer space). This count includes datagrams in ipForwDatagrams that meet this discard criterion.	read-only

Table 46. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipOutNoRoutes { ip 12 }	Counter	The number of IP datagrams discarded because no route can transmit them to their destination. This count includes packets in ipForwDatagrams that meet this no-route criterion.	read-only
ipReasmTimeout { ip 13 }	INTEGER	The maximum number of seconds that received fragments are held while awaiting reassembly at this entry.	read-only
ipReasmReqds { ip 14 }	Counter	The number of IP fragments that are received and need to be reassembled at this entry.	read-only
ipReasmOKs { ip 15 }	Counter	The number of IP datagrams reassembled without problems.	read-only
ipReasmFails { ip 16 }	Counter	The number of failures detected by the IP reassembly algorithm. This is not a count of discarded IP fragments because some algorithms can lose track of the number of fragments by combining them as they are received.	read-only
ipFragOKs { ip 17 }	Counter	The number of IP datagrams that have fragmented at this entry without problems.	read-only
ipFragFails { ip 18 }	Counter	The number of IP datagrams that should have been fragmented at this entry, but were not because their *Don't Fragment* flag was set.	read-only
ipFragCreates { ip 19 }	Counter	The number of IP datagram fragments that have been generated, because of fragmentation at this entry.	read-only
ipAddrTable { ip 20 }	SEQUENCE OF IpAddrEntry	A table that contains addressing information relevant to this entry's IP addresses.	read-only
ipAddrEntry { ipAddrTable 1 }	IpAddrEntry ::= SEQUENCE ipAdEntAddr IpAddress, ipAdEnt|f|ndex INTEGER, ipAdEntNetMask IpAddress, ipAdEntBcastAddr INTEGER ipAdEntReasmMaxSize INTEGER	The addressing information for one of this entry's IP addresses.	read-only
ipAdEntAddr { ipAddrEntry 1 }	IpAddress	The IP address pertaining to this entry's addressing information.	read-only

Table 46. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipAdEntIfIndex { ipAddrEntry 2 }	INTEGER	The index value that uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface that is identified by the same value of ifIndex.	read-only
ipAdEntNetMask { ipAddrEntry 3 }	IpAddress	The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the host bits set to 0.	read-only
ipAdEntBcastAddr { ipAddrEntry 4 }	INTEGER	The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the internet standard all-ones broadcast address is used, the value is 1.	read-only
ipAdEntReasmMaxSize { ipAddrEntry 5 }	INTEGER	The size of the largest IP datagram that this entity can reassemble from incoming IP fragmented datagrams received on this interface.	read-only
ipRoutingTable { ip 21 }	SEQUENCE OF IpRouteEntry	This entry's IP routing table.	read-write
ipRouteEntry { ipRoutingTable 1 }	IpRouteEntry ::= SEQUENCE ipRouteDest IpAddress, ipRouteIfIndex INTEGER, ipRouteMetric 1 INTEGER, ipRouteMetric 2 INTEGER, ipRouteMetric 3 INTEGER, ipRouteMetric 4 INTEGER, ipRouteNextHop IpAddress, ipRouteType INTEGER, ipRouteProto INTEGER, ipRouteAge INTEGER ipRouteMask INTEGER	A route to a particular destination.	read-write

Table 46. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipRouteDest { ipRouteEntry 1 }	IpAddress	The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple default routes can appear in the table, but access to these multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.	read-write
ipRouteIfIndex { ipRouteEntry 2 }	INTEGER	The index value that uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface that is identified by the same value of ifIndex.	read-write
ipRouteMetric1 { ipRouteEntry 3 }	INTEGER	The primary routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.	read-write[1]
ipRouteMetric2 { ipRouteEntry 4 }	INTEGER	An alternative routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.	read-write
ipRouteMetric3 { ipRouteEntry 5 }	INTEGER	An alternative routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.	read-write
ipRouteMetric4 { ipRouteEntry 6 }	INTEGER	An alternative routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.	read-write
ipRouteNextHop { ipRouteEntry 7 }	IpAddress	The IP address of the next hop of this route.	read-write
ipRouteType { ipRouteEntry 8 }	INTEGER other (1), invalid (2), direct (3), remote (4)	The type of route.	read-write

Table 46. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipRouteProto { ipRouteEntry 9 }	INTEGER other (1), local (2), netmgmt (3), icmp (4), egp (5), ggp (6), hello (7), rip (8), is-is (9), es-is (10), ciscoIgrp (11), bbnSpfIgp (12), ospf (13)	The routing mechanism by which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.	read-only
ipRouteAge { ipRouteEntry 10 }	INTEGER	The number of seconds since this route was last updated or otherwise determined to be correct. Note semantics of *too old* cannot be implied, except through knowledge of the routing protocol by which the route was learned.	read-write
ipRouteMask { ipRouteEntry 11 }	ipAddress	Indicate the mask to be logically ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belongs to a class–A, B, or C network. Then use one of the following: **mask network** **255.0.0.0** class-A **255.255.0.0** class-B **255.255.255.0** class-C If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. All IP routing subsystems implicitly use this mechanism.	read-write

IP Address Translation Table

Table 47 lists the objects in the IP Translation table.

Table 47. IP Address Translation Table

Object	Syntax	Definition	Access
ipNetToMediaTable { ip 22 }	SEQUENCE OF IpNetToMediaEntry	The IP Address Translation table used for mapping from IP addresses to physical addresses.	read-write[1]

Table 47. IP Address Translation Table (continued)

Object	Syntax	Definition	Access
IpNetToMediaEntry { ipNetToMediaTable 1 }	IpNetToMediaEntry ::= SEQUENCE ipNetToMediaIfIndex INTEGER, ipNetToMediaPhysAddress OCTET STRING, ipNetToMediaNetAddress IpAddress, ipNetToMediaType INTEGER	Each entry contains one IpAddress to physical address equivalence.	read-write
ipNetToMediaIfIndex { ipNetToMediaEntry 1 }	INTEGER	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.	read-write
ipNetToMediaPhysAddress { ipNetToMediaEntry 2 }	OCTET STRING	The media-dependent physical address.	read-write
ipNetToMediaNetAddress { ipNetToMediaEntry 3 }	IpAddress	The IpAddress corresponding to the media-dependent physical address.	read-write
ipNetToMediaType { ipNetToMediaEntry 4 }	INTEGER other(1), invalid(2), dynamic(3), static(4),	The type of mapping. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively disassociates the interface identified with the entry from the mapping identified with the entry. Whether the agent removes an invalidated entry from the table is an implementation-specific matter. Accordingly, management stations must be prepared to receive tabular information from agents that correspond to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.	read-write

ICMP Group

Table 48 lists the objects in the ICMP group. The ICMP objects are the input and output error and control message statistics for the IP layer.

Table 48. Implementation of the ICMP Group

Object	Syntax	Definition	Access
icmpInMsgs { icmp 1 }	Counter	The number of ICMP messages that the entry receives. This counter includes all those counted by icmpInErrors.	read-only
icmpInErrors { icmp 2 }	Counter	The number of ICMP messages that the entry receives and determines ICMP specific errors (bad ICMP checksums, bad length).	read-only

Table 48. Implementation of the ICMP Group (continued)

Object	Syntax	Definition	Access
icmpInDestUnreachs { icmp 3 }	Counter	The number of ICMP destination messages that cannot be reached.	read-only
icmpInTimeExcds { icmp 4 }	Counter	The number of ICMP destination messages that cannot be reached.	read-only
icmpInParmProbs { icmp 5 }	Counter	The number of ICMP Parameter Problem messages received.	read-only
icmpInSrcQuenchs { icmp 6 }	Counter	The number of ICMP Source Quench messages received.	read-only
icmpInRedirects { icmp 7 }	Counter	The number of ICMP Redirect messages received.	read-only
icmpInEchos { icmp 8 }	Counter	The number of ICMP Echo (request) messages received.	read-only
icmpInEchoReps { icmp 9 }	Counter	The number of ICMP Echo Reply messages received.	read-only
icmpInTimestamps { icmp 10 }	Counter	The number of ICMP Timestamp (request) messages received.	read-only
icmpInTimestampReps { icmp 11 }	Counter	The number of ICMP Timestamp Reply messages received.	read-only
icmpInAddrMasks { icmp 12 }	Counter	The number of ICMP Address Mask Request messages received.	read-only
icmpInAddrMaskReps { icmp 13 }	Counter	The number of ICMP Address Mask Reply messages received.	read-only
icmpOutMsgs { icmp 14 }	Counter	The number of ICMP messages sent. This counter includes icmpOutErrors.	read-only
icmpOutErrors { icmp 15 }	Counter	The number of ICMP messages that this entry did not send, because of problems within ICMP (for example, no buffers). This value should not include errors outside the ICMP layer (for example, the inability of IP to route the resulting datagram). In some implementations, there may not be error types that contribute to the counter's value.	read-only
icmpOutDestUnreachs { icmp 16 }	Counter	The number of ICMP Destination Unreachable messages sent.	read-only
icmpOutTimeExcds { icmp 17 }	Counter	The number of ICMP Time Exceeded messages sent.	read-only

Table 48. Implementation of the ICMP Group (continued)

Object	Syntax	Definition	Access
icmpOutParmProbs { icmp 18 }	Counter	The number of ICMP Parameter Problem messages sent.	read-only
icmpOutSrcQuenches { icmp 19 }	Counter	The number of ICMP Source Quench messages sent.	read-only
icmpOutRedirects { icmp 20 }	Counter	The number of ICMP Redirect messages sent. For a host, this object is always 0, because hosts do not send redirects.	read-only
icmpOutEchos { icmp 21 }	Counter	The number of ICMP Echo (request) messages sent.	read-only
icmpOutEchoReps { icmp 22 }	Counter	The number of ICMP Echo Reply messages sent.	read-only
icmpOutTimestamps { icmp 23 }	Counter	The number of ICMP Timestamp (request) messages sent.	read-only
icmpOutTimestampReps { icmp 24 }	Counter	The number of ICMP TimeStamp Reply messages sent.	read-only
icmpOutAddrMasks { icmp 25 }	Counter	The number of ICMP Address Mask Request messages sent.	read-only
icmpOutAddrMasksReps { icmp 26 }	Counter	The number of ICMP Address Mask Reply messages sent.	read-only

TCP Group

Table 49 lists the objects in the TCP group. The TCP objects are the data transmission statistics and connection data for the TCP layer.

Note: Objects that represent information about a particular TCP connection are transient; the objects exist only as long as the specified connection is in use.

Table 49. Implementation of the TCP Group

Object	Syntax	Definition	Access
tcpRtoAlgorithm { tcp 1 }	INTEGER other (1), none of the following constant (2), a constant rto rsre (3), MIL-STD-1778, Appendix B vanj (4) Van Jacobson's algorithm	The algorithm used to determine the time-out value used for retransmitting unacknowledged octets.	read-only

Table 49. Implementation of the TCP Group (continued)

Object	Syntax	Definition	Access
tcpRtoMin { tcp 2 }	INTEGER	The minimum value allowed by a TCP implementation for the retransmission time-out, measured in milliseconds. Semantics for objects of this type depend upon the algorithm used to determine the retransmission time-out. For example, when the time-out algorithm is rsre (3), an object of this type has the semantics of the LBOUND quantity.	read-only
tcpRtoMax { tcp 3 }	INTEGER	The maximum value allowed by a TCP implementation for the retransmission time-out, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission time-out. For example, when the time-out algorithm is rsre (3), an object of this type has the semantics of the UBOUND quantity.	read-only
tcpMaxConn { tcp 4 }	INTEGER	The limit on the number of TCP connections the entry can support. In entries where the maximum number of connections is dynamic, this object should be *-1*.	read-only
tcpActiveOpens { tcp 5 }	Counter	The number of TCP connections that have made a direct transition to the SYN-SENT state from the CLOSED state.	read-only
tcpPassiveOpens { tcp 6 }	Counter	The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.	read-only
tcpAttemptFails { tcp 7 }	Counter	The number of TCP connections that have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.	read-only
tcpEstabResets { tcp 8 }	Counter	The number of TCP connections that have made a direct transition to the CLOSED state from either the ESTABLISHED or CLOSE-WAIT state.	read-only
tcpCurrEstab { tcp 9 }	Gauge	The number of TCP connections of the current state that are either ESTABLISHED or CLOSE-WAIT.	read-only
tcpInSegs { tcp 10 }	Counter	The number of TCP segments including those received in error. This count includes segments received on established connections.	read-only
tcpOutSegs { tcp 11 }	Counter	The number of TCP segments sent including those on established connections, but excluding those containing only retransmitted octets.	read-only

Table 49. Implementation of the TCP Group (continued)

Object	Syntax	Definition	Access
tcpRetransSegs { tcp 12 }	Counter	The number of TCP segments retransmitted that contain one or more previously transmitted octets.	read-only
tcpConnTable { tcp 13 }	SEQUENCE OF TcpConnEntry	A table that contains TCP connnection-specific information.	read-only
tcpConnEntry	TcpConnEntry ::= SEQUENCE tcpConnState INTEGER, tcpConnLocalAddress IpAddress, tcpConnLocalPort INTEGER (0..65535), tcpConnRemAddress IpAddress, tcpConnRemPort INTEGER (0..65535)	Information about a certain current TCP connection. An object of this type is transient. It does not exist when (or soon after) the connection makes the transition to the CLOSED state.	read-only
tcpConnState { tcpConnEntry 1 }	INTEGER closed(1), listen(2), synSent(3), synReceived(4), established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9), closing(10), timeWait(11)	The TCP connection status.	read-only
tcpConnLocalAddress { tcpConnEntry 2 }	IpAddress	The local IP address of this TCP connection.	read-only
tcpConnLocalPort { tcpConnEntry 3 }	INTEGER (0..65535)	The local port number of this TCP connection.	read-only
tcpConnRemAddress { tcpConnEntry 4 }	IpAddress	The remote IP address of this TCP connection.	read-only
tcpConnRemPort { tcpConnEntry 5 }	INTEGER (0..65535)	The remote port number of this TCP connection.	read-only
tcpInErrs { tcp 14 }	Counter	The total number of segments received in error (for example, bad TCP checksums).	read-only
tcpOutRsts { tcp 15 }	Counter	The number of TCP segments sent containing the RST flag.	read-only

UDP Group

Table 50 lists the objects in the UDP group. The UDP objects are the datagram statistics of the UDP layer.

Table 50. Implementation of the UDP Group

Object	Syntax	Definition	Access
udpInDatagrams { udp 1 }	Counter	The number of UDP datagrams delivered to UDP users.	read-only
udpNoPorts { udp 2 }	Counter	The number of UDP datagrams received where there was no application at the destination port.	read-only
udpInErrors { udp 3 }	Counter	The number of UDP datagrams received that could not be delivered for reasons other than the lack of an application at the destination port.	read-only
udpOutDatagrams { udp 4 }	Counter	The number of UDP datagrams sent from this entry.	read-only
udpTable { udp 5 }	SEQUENCE of UDPEntry	Information about this ENTITY's UDP end-points on which a local application is currently accepting datagrams.	read-only
udpEntry { udpTable 1 }	UdpEntry ::= SEQUENCE udpLocalAddress IpAddress, udpLocalPort INTEGER	Information about a certain current UDP listener.	read-only
udpLocalAddress { udp Entry 1 }	IpAddress	The local IP address for this UDP listener. 0.0.0.0 is a listener which is willing to accept datagrams for any IP interface associated with the node.	read-only
udpLocalPort { udpEntry 2 }	INTEGER	The local port number for this UDP listener.	read-only

Bridge Group

"Bridge Group" lists the objects in the Bridge group.

Table 51. Implementation of the Bridge Group - dot1dBase

Object	Syntax	Definition	Access
dot1dBase {dot1dBase 1}	OBJECT IDENTIFIER	This mandatory group contains the objects which are applicable to all types of bridges. **Note for z/VM:** The virtual switch (VSWITCH) is the z/VM implementation of a bridge.	read-only

Table 51. Implementation of the Bridge Group - dot1dBase (continued)

Object	Syntax	Definition	Access
dot1dBaseBridgeAddress {dot1dBridge 1}	MacAddress	The MAC address used by this bridge when it must be referred to in a unique fashion. It is recommended that this be the numerically smallest MAC address of all ports that belong to this bridge. However, it is only required to be unique. When concatenated with dot1dStpPriority, a unique BridgeIdentifier is formed which is used in the Spanning Tree Protocol. **Note for z/VM:** This is set using SET VSWITCH MACID command or the MODIFY VSWITCH MACID configuration statement.	read-only
dot1dBaseNumPorts {dot1dBase 2}	INTEGER	The number of ports controlled by this bridging entity. **Note for z/VM:** This is the number of guest NICs active for the virtual switch plus the number of RDEVs active for the virtual switch.	read-only
dot1dBaseType {dot1dBase 3}	INTEGER unknown(1), transparent-only(2), sourceroute-only(3), srt(4)	Indicates what type of bridging this bridge can perform. If a bridge is actually performing a certain type of bridging, this will be indicated by entries in the port table for the given type. **Note for z/VM:** Virtual switches have a value of transparent-only (2).	read-only
dot1dBasePortTable {dot1dBase 4}	SEQUENCE OF dot1dBasePortEntry	A table that contains generic information about every port that is associated with this bridge. Transparent, sourceroute, and srt ports are included.	read-only
dot1dBasePortEntry {dot1dBasePortTable 1}	dot1dTpPortEntry := SEQUENCE dot1dBasePort INTEGER, dot1dBasePortIfIndex INTEGER, dot1dBasePortCircuit OBJECT IDENTIFER, dot1dBasePortDelayExceeded DiscardsCounter, dot1dBasePortMtuExceeded DiscardsCounter	A list of information for each port of this transparent bridge.	read-only
dot1dBasePort {dot1dBasePortEntry 1}	INTEGER (1..65535)	The port number of the port for which this entry contains bridge management information. **Note for z/VM:** The port number may change if a guest uncouples its NIC from the virtual switch. However, if the interface is stopped and restarted, the port number remains constant.	read-only

Table 51. Implementation of the Bridge Group - dot1dBase (continued)

Object	Syntax	Definition	Access
dot1dBasePortIfIndex {dot1dBasePortEntry 2}	INTEGER	The value of the instance of the ifIndex object, defined in MIB-II, for the interface corresponding to this port.	read-only
dot1dBasePortCircuit {dot1dBasePortEntry 3}	OBJECT IDENTIFIER	For a port which (potentially) has the same value of dot1dBasePortIfIndex as another port on the same bridge, this object contains the name of an object instance unique to this port. For a port which has a unique value of dot1dBasePortIfIndex, this object can have the value of { 0 0 }. **Note for z/VM:** This contains { 0 0 }.	read-only
dot1dBasePortDelay-ExceededDiscards {dot1dBasePortEntry 4}	Counter	The number of frames discarded by this port due to excessive transit delay through the bridge. It is incremented by both transparent and source route bridges. **Note for z/VM:** This contains a value of 0.	read-only
dot1dBasePortMtu-ExceededDiscards {dot1dBasePortEntry 5}	Counter	The number of frames discarded by this port due to an excessive size. It is incremented by both transparent and source route bridges. **Note for z/VM:** This contains a value of 0.	read-only
dot1dTp {dot1dBridge 4}	OBJECT IDENTIFIER		read-only
dot1dTpLearned-EntryDiscards {dot1dTp 1}	Counter	The total number of Forwarding Database entries, which have been or would have been learned, but have been discarded due to a lack of space to store them in the Forwarding Database. If this counter is increasing, it indicates that the Forwarding Database is regularly becoming full (a condition which has unpleasant performance effects on the subnetwork). If this counter has a significant value but is not presently increasing, it indicates that the problem has been occurring but is not persistent. **Note for z/VM:** This contains a value of 0.	read-only
dot1dTpAgingTime {dot1dTp 2}	INTEGER (10..1000000)	The timeout period in seconds for aging out dynamically learned forwarding information. 802.1D-1990 recommends a default of 300 seconds. **Note for z/VM:** This contains a value of 1000000.	read-only

Table 51. Implementation of the Bridge Group - dot1dBase (continued)

Object	Syntax	Definition	Access
dot1dTpFdbTable {dot1dTp 3}	SEQUENCE OF dot1dTpFdbEntry	A table that contains information about unicast entries for which the bridge has forwarding and/or filtering information. The information is used by the transparent bridging function in determining how to propagate a received frame.	read-only
dot1dTpFdbEntry {dot1dTpFdbTable 1}	dot1dTpFdbEntry := SEQUENCE dot1dTpFdbAddress dot1dTpFdbPort dot1dTpFdbStatus	Information about a specific unicast MAC address for which the bridge has some forwarding and/or filtering information. **Note for z/VM:** For a virtual switch the **IP** attribute, only one MAC address is known to the outside world — the MACID associated with the virtual switch network connection. Therefore only one instance of this information is returned for an **IP** (Layer 3) virtual switch. For an **ETHERNET** (Layer 2) virtual switch, information is returned for all guest ports and all ports associated with virtual switch RDEV connections to the external network.	read-only
dot1dTpFdbAddress {dot1dTpFdbEntry 1}	MacAddress	A unicast MAC address for which the bridge has forwarding and/or filtering information.	read-only
dot1dTpFdbPort {dot1dTpFdbEntry 2}	INTEGER	Either the value *0*, or the port number of the port on which a frame having a source address equal to the value of the corresponding instance of dot1dTpFdbAddress has been seen. A value of *0* indicates that the port number has not been learned but that the bridge does have some forwarding/filtering information about this address (for example, in the dot1dStaticTable). Implementors are encouraged to assign the port value of this object whenever it is learned, even for addresses for which the corresponding value of dot1dTpFdbStatus is not learned(3).	read-only

Table 51. Implementation of the Bridge Group - dot1dBase (continued)

Object	Syntax	Definition	Access
dot1dTpFdbStatus {dot1dTpFdbEntry 3}	INTEGER other (1), invalid (2), learned (3), self (4), mgmt (5)	The status of this entry. The meanings of the values are: **other(1)** none of the following. This would include the case where some other MIB object (not the corresponding instance of dot1dTpFdbPort, nor an entry in the dot1dStaticTable) is being used to determine if and how frames addressed to the value of the corresponding instance of dot1dTpFdbAddress are being forwarded. **invalid(2)** this entry is not longer valid (for example, it was learned but has since aged-out), but has not yet been flushed from the table. **learned(3)** the value of the corresponding instance of dot1dTpFdbPort was learned, and is being used. **self(4)** the value of the corresponding instance of dot1dTpFdbAddress represents one of the bridge's addresses. The corresponding instance of dot1dTpFdbPort indicates which of the bridge's ports has this address. **mgmt(5)** the value of the corresponding instance of dot1dTpFdbAddress is also the value of an existing instance of dot1dStaticAddress. **Note for z/VM:** This always has the value of learned (3).	read-only
dot1dTpPortTable {dot1dTp 4}	SEQUENCE OF dot1dTpPortEntry	A table that contains information about every port that is associated with this transparent bridge.	read-only
dot1dTpPortEntry {dot1dTpPortTable 1}	dot1dTpPortEntry := SEQUENCE dot1dTpPort INTEGER dot1dTpPortMaxInfo INTEGER dot1dTpPortInFrames Counter dot1dTpPortOutFrames Counter dot1dTpPortInDiscards Counter	A list of information for each port of this transparent bridge.	read-only
dot1dTpPort {dot1dTpPortEntry 1}	INTEGER (1..65535)	The port number of the port for which this entry contains transparent bridging management information.	read-only
dot1dTpPortMaxInfo {dot1dTpPortEntry 2}	INTEGER	The maximum size of the INFO (non-MAC) field that this port will receive or transmit.	read-only

Table 51. Implementation of the Bridge Group - dot1dBase (continued)

Object	Syntax	Definition	Access
dot1dTpPortInFrames {dot1dTpPortEntry 3}	Counter	The number of frames that have been received by this port from its segment. Note that a frame received on the interface corresponding to this port is only counted by this object if and only if it is for a protocol being processed by the local bridging function, including bridge management frames.	read-only
dot1dTpPortOutFrames {dot1dTpPortEntry 4}	Counter	The number of frames that have been transmitted by this port to its segment. Note that a frame transmitted on the interface corresponding to this port is only counted by this object if and only if it is for a protocol being processed by the local bridging function, including bridge management frames.	read-only
dot1dTpPortInDiscards {dot1dTpPortEntry 5}	Counter	Count of valid frames received which were discarded (that is, filtered) by the Forwarding Process.	read-only

Appendix F. SNMP Generic TRAP Types

This appendix lists the generic trap types that can be received by SNMP.

Value	Type	Description
0	coldStart	A coldStart trap signifies that the sending protocol entity is reinitializing itself so that the agent's configuration or the protocol entity implementation can be altered.
1	warmStart	A warmStart trap signifies that the sending protocol entity is reinitializing itself so that neither the agent configuration nor the protocol entity implementation can be altered.
2	linkDown	A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration. A Trap-PDU of type linkDown contains, as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.
3	linkUp	A linkUp trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up. A Trap-PDU of type linkUp contains, as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.
4	authenticationFailure	An authenticationFailure trap signifies that the sending protocol entity is the addressee of a protocol message that is not properly authenticated.
5	egpNeighborLoss	An egpNeighborLoss trap signifies that an EGP neighbor for whom the sending protocol entity was an EGP peer has been marked down and the peer relationship no longer exists. The Trap-PDU of the egpNeighborLoss contains, as the first element of its variable-bindings, the name and value of the egpNeighAddr instance for the affected neighbor.

© Copyright IBM Corp. 1987, 2016

SNMP Generic TRAP Types

Value	Type	Description
6	enterpriseSpecific	An enterpriseSpecific trap signifies that the sending protocol entity recognizes that some enterprise-specific event has occurred. The specific-trap field identifies the particular trap that occurred.

Appendix G. 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.

Many features of TCP/IP for z/VM are based on the following RFCs:

RFC	Title	Author
768	*User Datagram Protocol*	J.B. Postel
791	*Internet Protocol*	J.B. Postel
792	*Internet Control Message Protocol*	J.B. Postel
793	*Transmission Control Protocol*	J.B. Postel
821	*Simple Mail Transfer Protocol*	J.B. Postel
822	*Standard for the Format of ARPA Internet Text Messages*	D. Crocker
823	*DARPA Internet Gateway*	R.M. Hinden, A. Sheltzer
826	*Ethernet Address Resolution Protocol: or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware*	D.C. Plummer
854	*Telnet Protocol Specification*	J.B. Postel, J.K. Reynolds
856	*Telnet Binary Transmission*	J.B. Postel, J.K. Reynolds
857	*Telnet Echo Option*	J.B. Postel, J.K. Reynolds
877	*Standard for the Transmission of IP Datagrams over Public Data Networks*	J.T. Korb
885	*Telnet End of Record Option*	J.B. Postel
903	*Reverse Address Resolution Protocol*	R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
904	*Exterior Gateway Protocol Formal Specification*	D.L. Mills
919	*Broadcasting Internet Datagrams*	J.C. Mogul
922	*Broadcasting Internet Datagrams in the Presence of Subnets*	J.C. Mogul
950	*Internet Standard Subnetting Procedure*	J.C. Mogul, J.B. Postel
952	*DoD Internet Host Table Specification*	K. Harrenstien, M.K. Stahl, E.J. Feinler
959	*File Transfer Protocol*	J.B. Postel, J.K. Reynolds
974	*Mail Routing and the Domain Name System*	C. Partridge
1009	*Requirements for Internet Gateways*	R.T. Braden, J.B. Postel
1013	*X Window System Protocol, Version 11: Alpha Update*	R.W. Scheifler
1014	*XDR: External Data Representation Standard*	Sun Microsystems Incorporated
1027	*Using ARP to Implement Transparent Subnet Gateways*	S. Carl-Mitchell, J.S. Quarterman
1032	*Domain Administrators Guide*	M.K. Stahl

© Copyright IBM Corp. 1987, 2016

RFC	Title	Author
1033	*Domain Administrators Operations Guide*	M. Lottor
1034	*Domain Names—Concepts and Facilities*	P.V. Mockapetris
1035	*Domain Names—Implementation and Specification*	P.V. Mockapetris
1042	*Standard for the Transmission of IP Datagrams over IEEE 802 Networks*	J.B. Postel, J.K. Reynolds
1055	*Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP*	J.L. Romkey
1057	*RPC: Remote Procedure Call Protocol Version 2 Specification*	Sun Microsystems Incorporated
1058	*Routing Information Protocol*	C.L. Hedrick
1091	*Telnet Terminal-Type Option*	J. VanBokkelen
1094	*NFS: Network File System Protocol Specification*	Sun Microsystems Incorporated
1112	*Host Extensions for IP Multicasting*	S. Deering
1118	*Hitchhikers Guide to the Internet*	E. Krol
1122	*Requirements for Internet Hosts-Communication Layers*	R.T. Braden
1123	*Requirements for Internet Hosts-Application and Support*	R.T. Braden
1155	*Structure and Identification of Management Information for TCP/IP-Based Internets*	M.T. Rose, K. McCloghrie
1156	*Management Information Base for Network Management of TCP/IP-based Internets*	K. McCloghrie, M.T. Rose
1157	*Simple Network Management Protocol (SNMP),*	J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin
1179	*Line Printer Daemon Protocol*	The Wollongong Group, L. McLaughlin III
1180	*TCP/IP Tutorial,*	T. J. Socolofsky, C.J. Kale
1183	*New DNS RR Definitions* (Updates RFC 1034, RFC 1035)	C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris,
1187	*Bulk Table Retrieval with the SNMP*	M.T. Rose, K. McCloghrie, J.R. Davin
1198	*FYI on the X Window System*	R.W. Scheifler
1207	*FYI on Questions and Answers: Answers to Commonly Asked Experienced Internet User Questions*	G.S. Malkin, A.N. Marine, J.K. Reynolds
1208	*Glossary of Networking Terms*	O.J. Jacobsen, D.C. Lynch
1213	*Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II,*	K. McCloghrie, M.T. Rose
1215	*Convention for Defining Traps for Use with the SNMP*	M.T. Rose
1228	*SNMP-DPI Simple Network Management Protocol Distributed Program Interface*	G.C. Carpenter, B. Wijnen
1229	*Extensions to the Generic-Interface MIB*	K. McCloghrie
1267	*A Border Gateway Protocol 3 (BGP-3)*	K. Lougheed, Y. Rekhter
1268	*Application of the Border Gateway Protocol in the Internet*	Y. Rekhter, P. Gross
1269	*Definitions of Managed Objects for the Border Gateway Protocol (Version 3)*	S. Willis, J. Burruss
1293	*Inverse Address Resolution Protocol*	T. Bradley, C. Brown
1270	*SNMP Communications Services*	F. Kastenholz, ed.

RFC	Title	Author
1323	*TCP Extensions for High Performance*	V. Jacobson, R. Braden, D. Borman
1325	*FYI on Questions and Answers: Answers to Commonly Asked New Internet User Questions*	G.S. Malkin, A.N. Marine
1351	*SNMP Administrative Model*	J. Davin, J. Galvin, K. McCloghrie
1352	*SNMP Security Protocols*	J. Galvin, K. McCloghrie, J. Davin
1353	*Definitions of Managed Objects for Administration of SNMP Parties*	K. McCloghrie, J. Davin, J. Galvin
1354	*IP Forwarding Table MIB*	F. Baker
1387	*RIP Version 2 Protocol Analysis*	G. Malkin
1389	*RIP Version 2 MIB Extension*	G. Malkin
1393	*Traceroute Using an IP Option*	G. Malkin
1397	*Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol*	D. Haskin
1398	*Definitions of Managed Objects for the Ethernet-like Interface Types*	F. Kastenholz
1440	*SIFT/UFT:Sender-Initiated/Unsolicited File Transfer*	R. Troth
1493	*Definition of Managed Objects for Bridges*	E. Decker, P. Langille, A. Rijsinghani, K. McCloghrie
1540	*IAB Official Protocol Standards*	J.B. Postel
1583	*OSPF Version 2*	J.Moy
1647	*TN3270 Enhancements*	B. Kelly
1700	*Assigned Numbers*	J.K. Reynolds, J.B. Postel
1723	*RIP Version 2 — Carrying Additional Information*	G. Malkin
1738	*Uniform Resource Locators (URL)*	T. Berners-Lee, L. Masinter, M. McCahill
1813	*NFS Version 3 Protocol Specification*	B. Callaghan, B. Pawlowski, P. Stauback, Sun Microsystems Incorporated
1823	*The LDAP Application Program Interface*	T. Howes, M. Smith
2060	*IMAP Version 4 Protocol Specification*	M. Crispin
2460	*Internet Protocol, Version 6 (IPv6) Specification*	S. Deering, R. Hinden
2052	*A DNS RR for specifying the location of services (DNS SRV)*	A. Gulbrandsen, P. Vixie
2104	*HMAC: Keyed-Hashing for Message Authentication*	H. Krawczyk, M. Bellare, R. Canetti
2195	*IMAP/POP AUTHorize Extension for Simple Challenge/Response*	J. Klensin, R. Catoe, P. Krumviede
2222	*Simple Authentication and Security Layer (SASL)*	J. Myers
2247	*Using Domains in LDAP/X.500 Distinguished Names*	S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri
2251	*Lightweight Directory Access Protocol (v3)*	M. Wahl, T. Howes, S. Kille

RFC	Title	Author
2252	*Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*	M. Wahl, A. Coulbeck, T. Howes, S. Kille
2253	*Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*	M. Wahl, S. Kille, T. Howes
2254	*The String Representation of LDAP Search Filters*	T. Howes
2255	*The LDAP URL Format*	T. Howes, M. Smith
2256	*A Summary of the X.500 (96) User Schema for use with LDAPv3*	M. Wahl
2279	*UTF-8, a transformation format of ISO 10646*	F. Yergeau
2373	*IP Version 6 Addressing Architecture*	R. Hinden, S. Deering
2461	*Neighbor Discovery for IP Version 6 (IPv6)*	T. Narten, E. Nordmark, W. Simpson
2462	*IPv6 Stateless Address Autoconfiguration*	S. Thomson, T. Narten
2463	*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*	A. Conta, S. Deering
2710	*Multicast Listener Discovery (MLD) for IPv6*	S. Deering, W. Fenner, B. Haberman
2713	*Schema for Representing Java Objects in an LDAP Directory*	V. Ryan, S. Seligman, R. Lee
2714	*Schema for Representing CORBA Object References in an LDAP Directory*	V. Ryan, R. Lee, S. Seligman
2732	*Format for Literal IPv6 Addresses in URLs*	R. Hinden, B. Carpenter, L. Masinter
2743	*Generic Security Service Application Program Interface Version 2, Update 1*	J. Linn
2744	*Generic Security Service API Version 2 : C-bindings*	J. Wray
2820	*Access Control Requirements for LDAP*	E. Stokes, D. Byrne, B. Blakley, P. Behera
2829	*Authentication Methods for LDAP*	M. Wahl, H. Alvestrand, J. Hodges, R. Morgan
2830	*Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*	J. Hodges, R. Morgan, M. Wahl
2831	*Using Digest Authentication as a SASL Mechanism*	P. Leach, C. Newman
2849	*The LDAP Data Interchange Format (LDIF)*	G. Good
2873	*TCP Processing of the IPv4 Precedence Field*	X. Xiao, A. Hannan, V. Paxson, E. Crabble
3377	*Lightweight Directory Access Protocol (v3): Technical Specification*	J. Hodges, R. Morgan
3484	*Default Address Selection for Internet Protocol version 6 (IPv6)*	R. Draves
3513	*Internet Protocol Version 6 (IPv6) Addressing Architecture*	R. Hinden, S. Deering
4191	*Default Router Preferences and More-Specific Routes*	R. Draves, D. Thaler
4517	*LDAP Syntaxes and Matching Rules*	S. Legg
4523	*LDAP Schema Definitions for X.509 Certificates*	K. Zeilenga
5095	*Deprecation of Type 0 Routing Headers in IPv6*	J. Abley, P. Savola, G. Neville-Nei
5175	*IPv6 Router Advertisement Flags Option*	B. Haberman, R. Hinden
5722	*Handling of Overlapping IPv6 Fragments*	S. Krishnan
6946	*Processing of IPv6 "Atomic" Fragments*	F. Gont

RFC	Title	Author
6980	*Security Implications of IPv6 Fragmentation with IPv6*	F. Gont

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or on a subscription basis. Online copies are available using FTP from the NIC at `nic.ddn.mil`. Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

Where:

nnnn Is the RFC number.
TXT Is the text format.
PS Is the PostScript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC nnnn` for text versions or a subject line of `RFC nnnn.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil`. Information is also available at Internet Engineering Task Force (www.ietf.org).

Appendix H. Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this book.

AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Program Interface
APPC	Advanced Program-to-Program Communications
APPN	Advanced Peer-to-Peer Networking
ARP	Address Resolution Protocol
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
AUI	Attachment Unit Interface
BFS	Byte File System
BIOS	Basic Input/Output System
BNC	Bayonet Neill-Concelman
CCITT	Comite Consultatif International Telegraphique et Telephonique. The International Telegraph and Telephone Consultative Committee
CLIST	Command List
CMS	Conversational Monitor System
CP	Control Program
CPI	Common Programming Interface
CREN	Corporation for Research and Education Networking
CSD	Corrective Service Diskette
CTC	Channel-to-Channel
CU	Control Unit
CUA	Common User Access
DASD	Direct Access Storage Device
DBCS	Double Byte Character Set
DLL	Dynamic Link Library
DNS	Domain Name System
DOS	Disk Operating System
DPI	Distributed Program Interface
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EISA	Enhanced Industry Standard Adapter
ESCON	Enterprise Systems Connection Architecture
FAT	File Allocation Table
FTAM	File Transfer Access Management
FTP	File Transfer Protocol
FTP API	File Transfer Protocol Applications Programming Interface
GCS	Group Control System
GDDM	Graphical Data Display Manager
GDDMXD	Graphics Data Display Manager Interface for X Window System
GDF	Graphics Data File
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPL	Initial Program Load
ISA	Industry Standard Adapter
ISDN	Integrated Services Digital Network

© Copyright IBM Corp. 1987, 2016

ISO	International Organization for Standardization
IUCV	Inter-User Communication Vehicle
JES	Job Entry Subsystem
JIS	Japanese Institute of Standards
JCL	Job Control Language
LAN	Local Area Network
LAPS	LAN Adapter Protocol Support
LCS	IBM LAN Channel Station
LDAP	Lightweight Directory Access Protocol
LPQ	Line Printer Query
LPR	Line Printer Client
LPRM	Line Printer Remove
LPRMON	Line Printer Monitor
LU	Logical Unit
MAC	Media Access Control
Mbps	Megabits per second
MBps	Megabytes per second
MCA	Micro Channel Adapter
MIB	Management Information Base
MIH	Missing Interrupt Handler
MILNET	Military Network
MHS	Message Handling System
MTU	Maximum Transmission Unit
MVS	Multiple Virtual Storage
MX	Mail Exchange
NCP	Network Control Program
NDIS	Network Driver Interface Specification
NFS	Network File System
NIC	Network Information Center
NLS	National Language Support
NSFNET	National Science Foundation Network
OS/2	Operating System/2®
OSA	Open Systems Adapter
OSF	Open Software Foundation, Inc.
OSI	Open Systems Interconnection
OSIMF/6000	Open Systems Interconnection Messaging and Filing/6000
OV/MVS	OfficeVision/MVS
OV/VM	OfficeVision/VM
PAD	Packet Assembly/Disassembly
PC	Personal Computer
PCA	Parallel Channel Adapter
PDN	Public Data Network
PDU	Protocol Data Units
PING	Packet Internet Groper
PIOAM	Parallel I/O Access Method
POP	Post Office Protocol
PROFS	Professional Office Systems
PSCA	Personal System Channel Attach
PSDN	Packet Switching Data Network
PU	Physical Unit
PVM	Passthrough Virtual Machine
RACF	Resource Access Control Facility
RARP	Reverse Address Resolution Protocol
REXEC	Remote Execution
REXX	Restructured Extended Executor Language

RFC	Request For Comments
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RSCS	Remote Spooling Communications Subsystem
SAA	Systems Application Architecture®
SBCS	Single Byte Character Set
SFS	Shared File System
SLIP	Serial Line Internet Protocol
SMIL	Structure for Management Information
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOA	Start of Authority
SPOOL	Simultaneous Peripheral Operations Online
SQL	IBM Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TSO	Time Sharing Option
TTL	Time-to-Live
UDP	User Datagram Protocol
VGA	Video Graphic Array
VM	Virtual Machine
VMCF	Virtual Machine Communication Facility
VM/ESA	Virtual Machine/Enterprise System Architecture
VMSES/E	Virtual Machine Serviceability Enhancements Staged/Extended
VTAM	Virtual Telecommunications Access Method
WAN	Wide Area Network
XDR	eXternal Data Representation

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

© Copyright IBM Corp. 1987, 2016

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The

sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

Third Party Copyright Information

Portions of this product documentation and associated software pertaining to the TCP/IP RPC software are Copyright (C) 1984 Sun Microsystems, Inc.

```
*
* Sun RPC is a product of Sun Microsystems, Inc. and is provided for
* unrestricted use provided that this legend is included on all tape
* media and as a part of the software program in whole or part.  Users
* may copy or modify Sun RPC without charge, but are not authorized
* to license or distribute it to anyone else except as part of a product or
* program developed by the user.
*
* SUN RPC IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND
* INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY
* AND FITNESS FOR A PARTICULAR  PURPOSE, OR ARISING FROM A
* COURSE OF DEALING, USAGE OR TRADE PRACTICE.
*
* Sun RPC is provided with no support and without any obligation on the
* part of Sun Microsystems, Inc. to assist in its use, correction,
* modification or enhancement.
*
* SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH
* RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE
* SECRETS OR ANY PATENTS BY SUN RPC OR ANY PART THEREOF.
*
* In no event will Sun Microsystems, Inc. be liable for any lost revenue
* or profits or other special, indirect and consequential damages, even if
* Sun has been advised of the possibility of such damages.
*
* Sun Microsystems, Inc.
* 2550 Garcia Avenue
* Mountain View, California  94043
*/
/*      @(#)rpc.h 1.1 86/02/03 SMI      */
/*
* rpc.h, Just includes the billions of rpc header files necessary to
```

```
 * do remote procedure calling.
 *
 * Copyright (C) 1984, Sun Microsystems, Inc.
 */

#include "nfstypes.h"               /* some typedefs */
#include "in.h"

/* external data representation interfaces */
#include "nfsxdr.h"                 /* generic (de)serializer */

/* Client side only authentication */
#include "nfsauth.h"                /* generic authenticator (client side) */

/* Client side (mostly) remote procedure call */
#include "nfsclnt.h"                /* generic rpc stuff */

/* semi-private protocol headers */
#include "nfsrmsg.h"      /* protocol for rpc messages */
#include "nfsaunix.h"     /* protocol for unix style cred */

/* Server side only remote procedure callee */
#include "rpcsvc.h"                 /* service manager and multiplexer */
#include "nfssauth.h"     /* service side authenticator */
```

— —

Portions of this product documentation and associated software pertaining to NSLookup are Copyright (c) 1985, 1989 Regents of the University of California.

```
* Redistribution and use in source and binary forms are permitted provided
* that: (1) source distributions retain this entire copyright notice and
* comment, and (2) distributions including binaries display the following
* acknowledgement:  “This product includes software developed by the
* University of California, Berkeley and its contributors” in the
* documentation or other materials provided with the distribution and in
* all advertising materials mentioning features or use of this software.
* Neither the name of the University nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
```

This software and documentation is based in part on BSD Networking Software, Release 2 licensed from The Regents of the University of California. We acknowledge the role of the Computer Systems Research Group and the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley and the Other Contributors in its development.

THIS SOFTWARE IS PROVIDED “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

— —

Portions of this product documentation and associated software pertaining to GDDM for X Windows System are Copyright (c) as follows:

* (c) Copyright 1989, OPEN SOFTWARE FOUNDATION, INC.
* (c) Copyright 1989, DIGITAL EQUIPMENT CORPORATION, MAYNARD,
* MASS.
* (c) Copyright 1987, 1988, 1989 HEWLETT-PACKARD COMPANY
* (c) Copyright 1988 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
* ALL RIGHTS RESERVED
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE
* USED AND COPIED ONLY IN ACCORDANCE WITH THE TERMS OF
* SUCH LICENSE AND WITH THE INCLUSION OF THE ABOVE
* COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES
* THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
* TO ANY OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE
* SOFTWARE IS HEREBY TRANSFERRED. THE INFORMATION IN THIS
* SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD
* NOT BE CONSTRUED AS A COMMITMENT BY OPEN SOFTWARE
* FOUNDATION, INC. OR ITS THIRD PARTY SUPPLIERS.

OPEN SOFTWARE FOUNDATION, INC. AND ITS THIRD PARTY SUPPLIERS, ASSUME NO RESPONSIBILITY FOR THE USE OR INABILITY TO USE ANY OF ITS SOFTWARE . OSF SOFTWARE IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, AND OSF EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Notice: Notwithstanding any other lease or license that may pertain to, or accompany the delivery of, this computer software, the rights of the Government regarding its use, reproduction and disclosure are as set forth in Section 52.227-19 of the FARS Computer Software-Restricted Rights clause.

(c) Copyright 1989, Open Software Foundation, Inc. Unpublished - all rights reserved under the Copyright laws of the United States.

RESTRICTED RIGHTS NOTICE: Use, duplication, or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013.

Open Software Foundation, Inc.
11 Cambridge Center
Cambridge, MA 02142
(617)621-8700

(c) Copyright 1989, Open Software Foundation, Inc.
ALL RIGHTS RESERVED

Open Software Foundation is a trademark of The Open Software Foundation, Inc.
OSF is a trademark of Open Software Foundation, Inc.
OSF/Motif is a trademark of Open Software Foundation, Inc.
Motif is a trademark of Open Software Foundation, Inc.
DEC is a registered trademark of Digital Equipment Corporation
DIGITAL is a registered trademark of Digital Equipment Corporation
X Window System is a trademark of the Massachusetts Institute of Technology

Portions of this product documentation and associated software pertaining to MP Route software are (c) Copyright IBM Corp. 2001, Copyright (c) 1991, 1993 The Regents of the University of California, and Copyright (c) 1993 Digital Equipment Corporation.

```
/*
* ++Copyright++ 1991, 1993
* -
* Copyright (c) 1991, 1993
*     The Regents of the University of California.  All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the above copyright
*     notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*     notice, this list of conditions and the following disclaimer in the
*     documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
*     must display the following acknowledgment:
*       This product includes software developed by the University of
*       California, Berkeley and its contributors.
* 4. Neither the name of the University nor the names of its contributors
*     may be used to endorse or promote products derived from this software
*     without specific prior written permission.
*
* -
* Portions Copyright (c) 1993 by Digital Equipment Corporation.
*
* Permission to use, copy, modify, and distribute this software for any
* purpose with or without fee is hereby granted, provided that the above
* copyright notice and this permission notice appear in all copies, and that
* the name of Digital Equipment Corporation not be used in advertising or
* publicity pertaining to distribution of the document or software without
* specific, written prior permission.
```

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at IBM copyright and trademark information - United States (www.ibm.com/legal/us/en/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds 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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Online Privacy Statement Highlights at http://www.ibm.com/privacy and the IBM Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled “Cookies, Web Beacons and Other Technologies”, and the IBM Software Products and Software-as-a-Service Privacy Statement at http://www.ibm.com/software/info/product-privacy.

Glossary

For a list of z/VM terms and their definitions, see *z/VM: Glossary*.

The z/VM glossary is also available through the online z/VM HELP Facility, if HELP files are installed on your z/VM system. For example, to display the definition of the term "dedicated device", issue the following HELP command:

```
help glossary dedicated device
```

While you are in the glossary help file, you can do additional searches:

- To display the definition of a new term, type a new HELP command on the command line:

  ```
  help glossary newterm
  ```

  This command opens a new help file inside the previous help file. You can repeat this process many times. The status area in the lower right corner of the screen shows how many help files you have open. To close the current file, press the Quit key (PF3/F3). To exit from the HELP Facility, press the Return key (PF4/F4).
- To search for a word, phrase, or character string, type it on the command line and press the Clocate key (PF5/F5). To find other occurrences, press the key multiple times.

  The Clocate function searches from the current location to the end of the file. It does not wrap. To search the whole file, press the Top key (PF2/F2) to go to the top of the file before using Clocate.

© Copyright IBM Corp. 1987, 2016

Bibliography

See the following publications for additional information about z/VM. For abstracts of the z/VM publications, see *z/VM: General Information*, GC24-6193.

Where to Get z/VM Information

z/VM product documentation and other z/VM information is available in IBM Knowledge Center - z/VM (www.ibm.com/support/knowledgecenter/SSB27U).

You can also obtain z/VM product publications from IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss).

z/VM Base Library

Overview

- *z/VM: License Information*, GC24-6200
- *z/VM: General Information*, GC24-6193
- *z/VM: Glossary*, GC24-6195

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6246
- *z/VM: Migration Guide*, GC24-6201
- *z/VM: Service Guide*, GC24-6247
- *z/VM: VMSES/E Introduction and Reference*, GC24-6243

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6167
- *z/VM: CMS Planning and Administration*, SC24-6171
- *z/VM: Connectivity*, SC24-6174
- *z/VM: CP Planning and Administration*, SC24-6178
- *z/VM: Enabling z/VM for OpenStack (Support for OpenStack Liberty Release)*, SC24-6251
- *z/VM: Getting Started with Linux on z Systems*, SC24-6194
- *z/VM: Group Control System*, SC24-6196
- *z/VM: I/O Configuration*, SC24-6198
- *z/VM: Running Guest Operating Systems*, SC24-6228
- *z/VM: Saved Segments Planning and Administration*, SC24-6229
- *z/VM: Secure Configuration Guide*, SC24-6230
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6236
- *z/VM: TCP/IP Planning and Customization*, SC24-6238
- *z/OS and z/VM: Hardware Configuration Manager User's Guide*, SC34-2670

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6176
- *z/VM: Performance*, SC24-6208

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6166
- *z/VM: CMS Primer*, SC24-6172
- *z/VM: CMS User's Guide*, SC24-6173
- *z/VM: CP Commands and Utilities Reference*, SC24-6175
- *z/VM: System Operation*, SC24-6233
- *z/VM: TCP/IP User's Guide*, SC24-6240
- *z/VM: Virtual Machine Operation*, SC24-6241
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6244
- *z/VM: XEDIT User's Guide*, SC24-6245

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6162
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6163
- *z/VM: CMS Application Multitasking*, SC24-6164
- *z/VM: CMS Callable Services Reference*, SC24-6165
- *z/VM: CMS Macros and Functions Reference*, SC24-6168
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6179
- *z/VM: CPI Communications User's Guide*, SC24-6180
- *z/VM: Enterprise Systems Architecture/Extended Configuration Principles of Operation*, SC24-6192

© Copyright IBM Corp. 1987, 2016

- *z/VM: Language Environment User's Guide*, SC24-6199
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6202
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6203
- *z/VM: OpenExtensions Commands Reference*, SC24-6204
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6205
- *z/VM: OpenExtensions User's Guide*, SC24-6206
- *z/VM: Program Management Binder for CMS*, SC24-6211
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6220
- *z/VM: REXX/VM Reference*, SC24-6221
- *z/VM: REXX/VM User's Guide*, SC24-6222
- *z/VM: Systems Management Application Programming*, SC24-6234
- *z/VM: TCP/IP Programmer's Reference*, SC24-6239
- *Common Programming Interface Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821
- *z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS*, SA76-0169
- *z/OS: Language Environment Concepts Guide*, SA22-7567
- *z/OS: Language Environment Debugging Guide*, GA22-7560
- *z/OS: Language Environment Programming Guide*, SA22-7561
- *z/OS: Language Environment Programming Reference*, SA22-7562
- *z/OS: Language Environment Run-Time Messages*, SA22-7566
- *z/OS: Language Environment Writing Interlanguage Communication Applications*, SA22-7563
- *z/OS MVS Program Management: Advanced Facilities*, SA23-1392
- *z/OS MVS Program Management: User's Guide and Reference*, SA23-1393

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6161
- *z/VM: CP Messages and Codes*, GC24-6177
- *z/VM: Diagnosis Guide*, GC24-6187
- *z/VM: Dump Viewing Facility*, GC24-6191
- *z/VM: Other Components Messages and Codes*, GC24-6207
- *z/VM: TCP/IP Diagnosis Guide*, GC24-6235
- *z/VM: TCP/IP Messages and Codes*, GC24-6237
- *z/VM: VM Dump Tool*, GC24-6242
- *z/OS and z/VM: Hardware Configuration Definition Messages*, SC34-2668

z/VM Facilities and Features

Data Facility Storage Management Subsystem for VM

- *z/VM: DFSMS/VM Customization*, SC24-6181
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6182
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6183
- *z/VM: DFSMS/VM Planning Guide*, SC24-6184
- *z/VM: DFSMS/VM Removable Media Services*, SC24-6185
- *z/VM: DFSMS/VM Storage Administration*, SC24-6186

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6188
- *z/VM: Directory Maintenance Facility Messages*, GC24-6189
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6190

Open Systems Adapter/Support Facility

- *Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935
- *Open Systems Adapter-Express Integrated Console Controller User's Guide*, SA22-7990
- *Open Systems Adapter-Express Integrated Console Controller 3215 Support*, SA23-2247
- *Open Systems Adapter-Express3 Integrated Console Controller Dual-Port User's Guide*, SA23-2266

Performance Toolkit for VM

- *z/VM: Performance Toolkit Guide*, SC24-6209
- *z/VM: Performance Toolkit Reference*, SC24-6210

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6212
- *z/VM: RACF Security Server Command Language Reference*, SC24-6213

- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6214
- *z/VM: RACF Security Server General User's Guide*, SC24-6215
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6216
- *z/VM: RACF Security Server Messages and Codes*, GC24-6217
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6218
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6219
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6231

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6223
- *z/VM: RSCS Networking Exit Customization*, SC24-6224
- *z/VM: RSCS Networking Messages and Codes*, GC24-6225
- *z/VM: RSCS Networking Operation and Use*, SC24-6226
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6227

Prerequisite Products

Device Support Facilities

- *Device Support Facilities: User's Guide and Reference*, GC35-0033

Environmental Record Editing and Printing Program

- *Environmental Record Editing and Printing Program (EREP): Reference*, GC35-0152
- *Environmental Record Editing and Printing Program (EREP): User's Guide*, GC35-0151

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM V6.4 library, that you may find helpful.

- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *TCP/IP Illustrated, Volume 1: The Protocols*, SR28-5586
- *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture*, SC31-6144
- *Internetworking With TCP/IP Volume II: Implementation and Internals*, SC31-6145
- *Internetworking With TCP/IP Volume III: Client-Server Programming and Applications*, SC31-6146
- *DNS and BIND in a Nutshell*, SR28-4970
- "MIB II Extends SNMP Interoperability," C. Vanderberg, *Data Communications*, October 1990.
- *"Network Management and the Design of SNMP,"* J.D. Case, J.R. Davin, M.S. Fedor, M.L. Schoffstall.
- *"Network Management of TCP/IP Networks: Present and Future,"* A. Ben-Artzi, A. Chandna, V. Warrier.
- "Special Issue: Network Management and Network Security,"*ConneXions-The Interoperability Report*, Volume 4, No. 8, August 1990.
- *The Art of Distributed Application: Programming Techniques for Remote Procedure Calls*, John R. Corbin, Springer-Verlog, 1991.
- *The Simple Book: An Introduction to Management of TCP/IP-based Internets*, Marshall T Rose, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Index

Special characters

~= filter 174

A

abbreviations and acronyms 503
ACCT (FTP subcommand) 58
address fields
  class A 14
  class B 14
  class C 14
  class D 14
Address Resolution Protocol (ARP) 7
AIX files 449
ALL (NETSTAT parameter) 187, 202
ALLCONN (NETSTAT parameter) 187, 203
AO (TELNET subcommand) 132
APL2 Character Set 455
APPEND (FTP subcommand) 59
application layer 5
applications, functions and protocols
  Domain Name System (DNS) 10, 395
  File Transfer Protocol (FTP) 9, 27
  GDDMXD 11
  MPRoute 11
  Network File System (NFS) 12, 297
  Remote Execution Protocol (REXEC) 12, 329
  Remote Printing (LPR) 11, 335
  Remote Procedure Call (RPC) 12
  Simple Mail Transfer Protocol (SMTP) 10
  Simple Network Management Protocol (SNMP) 10, 375
  Socket interfaces 12
  Telnet Protocol 9, 129
  Trivial File Transfer Protocol (TFTP) 111
  X Toolkit 11
  X Window System 11
approximate filter 174
ARP (NETSTAT parameter) 204
AS 400 files 452
ASCII (FTP subcommand) 59
ASCII mode
  with the TFTP GET subcommand 112
  with the TFTP MODE subcommand 114
ASCII-to-EBCDIC table 442
Athena widget 11
attacks
  Fraggle 193
  Ping-o-Death 193
  Smurf 193
attacks, denial-of-service (DOS)
  example 213
AYT (TELNET subcommand) 132

B

BINARY (FTP subcommand) 60
bit mask 16
BLOCK (NETSTAT parameter) 187
bridge 2
broadcast address format 15

C

CCC (FTP subcommand) 61
CD (FTP subcommand) 62
certificate
  removing 268
  self-signed, creating 253
Certificate (Key) Database Administration
  creating and using a key database 291
certificate management 233
changing the FTP file transfer type
  to ASCII 59
  to Image 60
  to Kanji 74, 96
character sets 437
client 2
CLIENTS (NETSTAT parameter) 188, 205
CLOSE (FTP subcommand) 66
CMS
  running LDAP client utilities from 137
CMS (FTP subcommand) 66
CMS command interface 434
CMSRESOL 433
code pages 437
command-line utilities
  ldapadd 156
  LDAPCHPW 146
  ldapcompare 149
  ldapdelete 153
  ldapmodify 156
  ldapmodrdn 169
  ldapsearch 174
  using 138
computer networks
  LAN 1
  WAN 1
CONN (NETSTAT parameter) 190, 209
Controlling File Translation (FTP) 51
CONVXLAT command 447
CP (NETSTAT parameter) 190, 210
CP SMSG command 318, 319, 320
customizing
  DBCS translation tables 443
  SBCS translation tables 442

D

data set names 449
datagram 2
datagram socket 376
DBCS facilities 461
debug
  levels 143
DEBUG (FTP subcommand) 67
DELARP (NETSTAT parameter) 190
DELETE (FTP subcommand) 68
deleting CMS record-length fields 326
DELIMIT (FTP subcommand) 68
DELNeighbor (NETSTAT parameter) 191
denial-of-service (DOS) attacks
  example 213

© Copyright IBM Corp. 1987, 2016

denial-of-service (DOS) attacks *(continued)*
Fraggle 193
Ping-o-Death 193
Smurf 193
DEVLINKS (NETSTAT parameter) 191, 211
diagnostic information 293
DiG command 422
DIG Command 422
DiG internal state information 422
DiG options 426
DiG program 421
DIR (FTP subcommand) 69
direct routing 13
domain name servers 396
Domain Name System (DNS)
CMS command interface to the resolver 434
CMSRESOL, resolver and name server 433
NSLOOKUP examples 419
Overview of Domain Name System 10, 395
query types 435
Querying Name Servers-DiG 421
Querying Name Servers-NSLOOKUP 401
resolvers 397
resource records 398
Using the Domain Name System 395
domain names 395
DOS names 451
dotted-decimal notation 183
DROP (NETSTAT parameter) 194, 213

E

EBCDIC (FTP subcommand) 70
EBCDIC-to-ASCII table 443
electronic mail
delivery 120
gateway 119
notes
non-delivery note 121
unknown recipient note 122
notes without PROFS interface 124
PROFS interface 122
SMTPQUEU command 120
TCP/IP recipients 122
Elliptic Curve Cryptographic (ECC) 273
Elliptic Curve Cryptography (ECC) support 235
EUCKANJI (FTP subcommand) 71
Extended Mail, PROFS 122

F

file name translation (NFS)
special names 325
file names 449
file naming format, FTP 46
File Transfer Protocol (FTP) 9, 27, 110
file transfer type
ASCII 51, 59, 100, 101
EBCDIC 51, 70, 100, 101
Image 51, 60, 100, 101
Kanji 51, 100, 101
filter
approximate 174
syntax 145
using for search 174
FILTERS (NETSTAT parameter) 194
foreign host 2
Fraggle attack 193
FTP
file transfer methods 51
naming files 46
FTP command format 28
FTP DATA File 31
Configuration Statements 33
CCONNTIME Statement 33
DATACTTIME Statement 34
DCONNTIME Statement 35
EPSV4 Statement 36
FTPKEEPALIVE Statement 37
INACTTIME Statement 38
LOADDBCSTABLE Statement 39
MYOPENTIME Statement 40
Statement Syntax 32
FTP data transfer methods 51
FTP EXEC interface 105
FTP EXIT return codes 105
FTP Kanji support 461
FTP parameters
EXIT 29
foreign host 28, 29
port number 28
TIMEOUT 29
TRACE 30
TRANSLATE 30
FTP reply codes 107
FTP servers, RACF support 110
FTP subcommands 61
ACCT 58
APPEND 59
ASCII 59
BINARY 60
CD 62, 64
CDUP 65
CLOSE 66
CMS 66
DEBUG 67
DELETE 68
DELIMIT 68
DIR 69
EBCDIC 70
EUCKANJI 71
GET 71
HELP 73
IBMKANJI 73
JIS78KJ 74
JIS83KJ 74
LCD 75
LOCSITE 76
LOCSTAT 77
LPWD 78
LS 78
MDELETE 80
MGET 81
MKDIR 81
MODE 82
MPUT 83
NETRC 84
NOOP 84
OPEN 85
PASS 86
PASSIVE 87
PUT 88
PWD 89

FTP subcommands *(continued)*
QUIT 90
QUOTE 90
RENAME 91
SENDPORT 92
SENDSITE 93
SITE 93
SIZE 96
SJISKANJI 96
STATUS 97
STRUCT 97
SUNIQUE 98
SYSTEM 99
TRACE 30
TRANSLATE 30
TYPE 100
USER 101

G

GATE (NETSTAT parameter) 194, 213
gateway 2
gddmx*ANFont command 367
gddmx*CMap: command 364
gddmx*Compr: command 366
gddmx*Enter: command 368
gddmx*GColornn: command 362
gddmx*Geometry: command 361
gddmx*GMCPnn: command 363
gddmx*HostRast: command 365
gddmx*NewLine: command 369
gddmx*PDTrace: command 367
gddmx*XClConn: command 366
gddmx*XSync: command 364
gddmx*ZWL: command 363
GDDMXD 11, 357, 375, 455
GDDMXD command 358
GDDMXD/VM
CMS GLOBALV command format 359
command format 358
demonstration programs
GXDEMO1 373
GXDEMO2 373
GXDEMO3 374
GXDEMO4 374
GXDEMO5 374
GXDEMO6 374
GXDEMO7 374
GDXAPLCS MAP 371
keyboard functions 369, 370
limitations 357
overview 357
problem determination 371, 372
trace 371, 372
user-specified options
CMap 364
GColornn 362
Geometry 361
GMCPnn 363
HostRast 365
XClConn 366
XSync 364
ZWL 363
VM/XA considerations 358
X DEFAULTS 360
GET (FTP subcommand) 71
GET (TFTP subcommand) 112
GLOBALV command 359
GSKKYMAN
using 233
gskkyman utility
being your own certificate authority 282
certificate, self signed
creating 253
certificates
removing 268
private key
removing 268
setting NLSPATH environment variable 234
using 233
GSKTRACE
setting NLSPATH environment variable 293
GSKTRACE command 294
gsktrace utility
setting NLSPATH environment variable 293

H

HELP (FTP subcommand) 73
HELP (NETSTAT subcommand) 195, 214
HELP (TELNET subcommand) 133
HELP (TFTP subcommand) 113
HOME (NETSTAT parameter) 196, 215
host
foreign 2
local 2

I

IBMKANJI (FTP subcommand) 73
IDENTIFY (NETSTAT subcommand) 196, 217
IMAP
using 124
indirect routing 13
Integrated Services Digital Network (ISDN) 1
internal error codes 109
internet address 2
internet addressing
broadcast address format 15
internet address 2
local address 13
network address format 13
network number 13
subnetwork address format 15
subnetwork number 13
Internet Control Message Protocol (ICMP) 7
Internet Environment
bridge 2
client 2
datagram 2
foreign host 2
gateway 2
host 2
internet address 2
local host 2
logical network 2
mapping 2
network 2
packet 2
physical network 1
port 2
protocol 2
router 2

Internet Environment *(continued)*
server 2
user 2
Internet Message Access Protocol (IMAP) 10
Internet Protocol (IP) 7
internetwork layer 5
internetwork protocols 7
Address Resolution Protocol (ARP) 7
Internet Control Message Protocol (ICMP) 7
Internet Protocol (IP) 7
INTERVAL (NETSTAT parameter) 197, 218
intrinsics 11
inverse query 435
IP (TELNET subcommand) 133
IUCV cross-memory facility 376

J

JIS78KJ 74
JIS83KJ 74

K

Kanji
EUCKANJI 71
IBMKANJI 73
JIS78KJ 74
JIS83KJ 74
SJISKANJI 96
support for FTP 461
Kanji, using 461
key management 233

L

layered architecture
application layer 5
internetwork layer 5
network layer 5
transport layer 5
LDAP
client utilities 137
LDAP URLs 144
LDAPADD
description 156
ldapadd utility
description 156
LDAPCHPW utility
description 146
LDAPCMPR
description 149
ldapcompare utility
description 149
ldapdelete utility
description 153
LDAPDLET
description 153
LDAPMDFY
description 156
ldapmodify utility
description 156
ldapmodrdn utility
description 169
LDAPMRDN
description 169
ldapsearch utility
description 174
LDAPSRCH
description 174
LEVEL (NETSTAT parameter) 198
licensing agreement 509
line mode 131
local address 13
Local Area Network (LAN) 1
local host 2
LOCSTAT (FTP subcommand) 77
LOCSTAT (TFTP subcommand) 113
logical network 2
LPQ
command 353
examples 354
usage 354
LPQ Command 353
LPR
command 339
examples 349
usage 346
LPR Command 339
LPRM
command 355
examples 356
usage 356
LPRSET
command 335
examples 338
usage 337
LPRSET Command 335
LS (FTP subcommand) 78

M

mail, electronic 119, 124
Management Information Base (MIB) objects 467
managing PKI private keys and certificates 233
mapping 2
mapping values (APL2) 455
MAXPKT (TFTP subcommand) 114
MDELETE (FTP subcommand) 80
message examples, notation used in xviii
MGET (FTP subcommand) 81
MIB/Network elements
Address Translation group 478
ICMP group 484
Interfaces group 473
IP Address Translation table 483
IP group 478
System group 471
TCP group 486
UDP group 489
MODE (FTP subcommand) 82
MODE (TFTP subcommand) 114
monitoring the TCP/IP network 183, 227
MOUNT (NFS subcommand) 298
MOUNTPW (NFS subcommand) 306
MPRoute 11
MPUT (FTP subcommand) 83
MVS data sets 450

N

name translation, NFS 324
NEighbor (NETSTAT parameter) 198
NETRC (FTP subcommand) 84
NETRC DATA File
  FTP 31
  how to use 453
  REXEC 331
NETSTAT 183, 224
  LEVEL 198
NETSTAT address interpretation 183
NETSTAT command format 186
NETSTAT examples 202
NETSTAT parameters
  ALL 187
  ALLCONN 187
  ARP 187
  BLOCK 187
  CLIENTS 188
  CONN 190
  CP 190
  DELARP 190
  DELNeighbor 191
  DEVLINKS 191
  DROP 194
  FILTERS 194
  GATE 194
  HELP 195
  HOME 196
  IDENTIFY 196
  INTERVAL 197
  NEighbor 198
  POOLSIZE 200
  RESETPOOL 200
  SOCKET 201
  TELNET 202
  UNBLOCK 202
  UP 202
NetView Command List Language 375
network
  logical 2
  physical 2
  STATUS 183
network address format 13
network layer 5
network management 375
network number 13
network protocols 6
NFS 297, 326
NFS CMS SMSG command 318
NFS commands
  Error Messages 298, 300, 311
  MOUNT 298, 300
  MOUNTPW 298, 300, 306
  UMOUNT 298, 300
NFS name translation
  special file names 325
NFS overview 12
NFS PC-NFS client 325
NFS servers, RACF support 326
NFS system return codes 311
NLSPATH environment variable, setting 234, 293
NOOP (FTP subcommand) 84
notation used in message and response examples xviii
NSLOOKUP command 401
  command line 401
  interactive session 401
NSLOOKUP interactive subcommands 404
  ? 406
  exit 404
  finger 405
  help 406
  ls 406
  lserver 407
  root 408
  server 408
  view 409
NSLOOKUP internal state information 401
NSLOOKUP options 409
NSLOOKUP program 401
NSLOOKUP set subcommands 409
  all 410
  class 410
  d2 411
  debug 411
  defname 412
  domain 412
  ignoretc 413
  nod2 411
  nodebug 411
  nodefname 412
  noignoretc 413
  norecurse 415
  nosearch 417
  novc 418
  port 414
  querytype 414
  recurese 415
  retry 415
  root 416
  search 417
  srchlist 417
  timeout 418
  type 414
  vc 418

O

OBEY (NETSTAT parameter) 199
obtaining information from the network 183
obtaining SSL trace information 294
octet mode
  with the TFTP GET subcommand 112
  with the TFTP MODE subcommand 114
OfficeVision 119
OPEN (FTP subcommand) 85
OPEN (TFTP subcommand) 115
Open System Interconnection (OSI) 1
OS/2 files 451
OSAINFO
  example 220
OSF/Motif 11

P

PA1 (TELNET subcommand) 134
packet 2
packet statistics 193
partitioned data set names 450
PASS (FTP subcommand) 86
PASSIVE (FTP subcommand) 87
password use with FTP
  ACCT 58

password use with FTP *(continued)*
PASS 86
QUOTE 90
USER 102
physical network 2
PING 228
PING command format 228
PING options
COUNT 229
LENGTH 229
TIMEOUT 229
Ping-o-Death attack 193
PKCS #12 files 235
POOLSIZE (NETSTAT parameter) 200, 221
port 2
private keys
removing 268
PROFS interface 123
protocol data unit (PDU) 376
protocols
definition 2
internetwork protocols 7
network protocols 6
PUT (FTP subcommand) 88
PUT (TFTP subcommand) 115
PWD (FTP subcommand) 89

Q

query engine 376
query options 424
query types
inverse query 435
standard query 435
querying name servers
DiG 421
NSLOOKUP 401
QUIT (FTP subcommand) 90
QUIT (TELNET subcommand) 134
QUIT (TFTP subcommand) 116
QUOTE (FTP subcommand) 90

R

RACF 110, 326
raw socket 376
RDN (relative distinguished name)
modifying 169
related protocols 497
Remote Execution Protocol (REXEC) 12, 329, 335
remote printing 11, 335, 356
Remote Procedure Call (RPC) 12
removing
certificate/private key from key database 268
RENAME (FTP subcommand) 91
RESETPOOL (NETSTAT parameter) 200, 222
resolvers 397
resource records 398
response examples, notation used in xviii
REXEC command format 329
REXMIT (TFTP subcommand) 116
router 2
routing
direct 13
indirect 13
RPCINFO 224
RPCINFO command format 225
RPCINFO parameters 224

S

search filter, syntax 145
Secure Socket Layer (SSL) 12
certificate (key) database administration 291
certificate management 293
creating and using a key database 292
obtaining a certificate 292
using self-signed certificates 293
SENDPORT (FTP subcommand) 92
SENDSITE (FTP subcommand) 93
sequential data set names 450
server 2
SET subcommand 409
setting
NLSPATH environment variable 234, 293
Simple Mail Transfer Protocol (SMTP) 10
Using DBCS with Mail 465
Simple Network Management Protocol (SNMP) 10
SITE (FTP subcommand) 93
SIZE (FTP subcommand) 96
SJISKANJI 96
SMSG (CMS subcommand) 318
SMSG interface to SMTP 125
SMTPQUEU 120
SMTPQUEU Command 120
Smurf attack 193
SNMP
command 377
command processor 376
generic TRAP types 495
major and minor error codes 393
managing an internet environment 375
MIB/Network elements 470
overview 10
Protocol 375
Query Engine Host Lookup 394
return codes 377
sample command lists 375
value types
SNMP GET 378, 390
SNMP GETNEXT 380
SNMP MIBVNAME 387
SNMP PING 388
SNMP SET 382
SNMP TRAPSOFF 386
SNMP TRAPSON 384
SNMP command processor 376
SNMP generic TRAP types 495
SNMP Get command 380
SNMP GET command 378
SNMP MIBVNAME command 387, 388
SNMP Set command 382
SNMP TRAPSOFF command 386
SNMP TRAPSON command 384
SNMP/Netview overview 376
SNMPIUCV task 376
SNMPMGMT command 375
SNMPRUN command 375
SNMPTRAP command 390
SOCKET (NETSTAT parameter) 201, 222
socket interfaces 12
sockets
datagram 376

sockets *(continued)*
raw 376
stream 376
SSL
obtaining trace information 294
SSL (Secure Socket Layer)
certificate (key) database administration 291
certificate management 293
creating and using a key database 292
obtaining a certificate 292
using self-signed certificates 293
standard query 435
STATUS (FTP subcommand) 97
stream socket 376
STRUCT (FTP subcommand) 97
Structure and Identification of Management Information (SMI) 467
subnetwork address format 15
subnetwork number 13
SUNIQUE (FTP subcommand) 98
SYNCH (TELNET subcommand) 135
syntax diagrams, how to read xvi
SYSTEM (FTP subcommand) 99

T

TCP/IP functional groups
application layer 5
internetwork layer 5
network layer 5
transport layer 5
TCP/IP protocols and functions 5
TELNET 129, 135
TELNET (NETSTAT subcommand) 202, 223
TELNET command format 129
TELNET control characters in line mode 135
TELNET function keys 131
TELNET modes
line mode 131
transparent mode 131
TELNET parameters
FIREWALL 130
LINEMODE 130
RECORD 130
TRANSLATE 130
TELNET PF key functions 131
Telnet Protocol 9
TELNET subcommands
AO 132
AYT 132
BRK 133
HELP 133
IP 133
PA1 134
QUIT 134
SYNCH 135
TELNET support display stations 131
TFTP 111, 116
TFTP command format 111
TFTP parameters
ASCII 114
OCTET 114
TRANSLATE 111
TFTP subcommands
GET 112
HELP 113
LOCSTAT 113
TFTP subcommands *(continued)*
MAXPKT 114
MODE 114
OPEN 115
PUT 115
QUIT 116
REXMIT 116
TRACE 116
trace
debug levels 143
TRACE (TFTP subcommand) 116
traceroute function (TRACERTE) 225
translation tables
Chinese DBCS 446
converting to binary 446
Hangeul DBCS 445
IBM 440
Japanese Kanji DBCS 444
search order 438
translation tables for TFTP 112
Transmission Control Protocol (TCP) 9
transparent mode 131
transport layer 5
transport protocols
Transmission Control Protocol (TCP) 9
User Datagram Protocol (UDP) 9
Trivial File Transfer Protocol (TFTP) 111
TYPE (FTP subcommand) 100

U

UNBLOCK (NETSTAT parameter) 202
UNIX syntax 71
UP (NETSTAT subcommand) 202, 224
user 2
USER (FTP subcommand) 101
User Datagram Protocol (UDP) 9
using translation tables 437
utility
ldapadd 156
LDAPCHPW 146
ldapcompare 149
ldapdelete 153
ldapmodify 156
ldapmodrdn 169
ldapsearch 174

V

virtual network 1

W

Wide Area Network (WAN) 1
widgets
Athena 11
OSF/Motif 11
Windows files 451
write logic for file mode 6 316

X

X-Toolkit overview 11
X-Windows overview 11

Product Number: 5741-A07

Printed in USA

SC24-6240-07