

z/VM



TCP/IP User's Guide

version 5 release 2

z/VM



TCP/IP User's Guide

version 5 release 2

Note:

Before using this information and the product it supports, read the information under “Notices” on page 381.

Second Edition (December 2005)

This edition applies to version 5, release 2, modification 0 of IBM z/VM (product number 5741-A05) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-6127-00.

© **Copyright International Business Machines Corporation 1987, 2005. All rights reserved.**

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

Contents

About This Book	xi
Who Should Read This Book	xi
What You Should Know before Reading This Book	xi
How to Use This Book	xi
How to Read Syntax Diagrams	xi
How the Term “internet” Is Used in This Book	xiv
How Numbers Are Used in This Book	xiv
What This Book Contains	xiv
Where to Find More Information	xvi
How to Send Your Comments to IBM	xvi
 Summary of Changes	xvii
SC24-6127-01, z/VM Version 5 Release 2	xvii
Changes to the NETSTAT Command	xvii
SC24-6127-00, z/VM Version 5 Release 1	xvii
Internet Protocol Version 6 (IPv6) Support	xvii
Enabled for LibraryCenter Advanced Search.	xvii
SC24-6020-02, z/VM Version 4 Release 4	xvii
TCP/IP Stack Security	xvii
Virtual Switch Support	xvii
SC24-6020-01, z/VM Version 4 Release 3	xviii
PASV FTP client support.	xviii
TCP/IP Stack Vulnerability Reduction	xviii
 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	4
Link Protocols.	6
Network Protocols	6
Transport Protocols.	8
Applications and Protocols	9
Routing.	13
Internet Addressing	14
IPv4 Addressing	14
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	31
FTP DATA File Statements	31
CCONNTIME Statement	32
DATACTTIME Statement	33
DCONNTIME Statement	34
INACTTIME Statement	35
LOADDBCSTABLE Statement	36
MYOPENTIME Statement	37
FTP Subcommands	38
Continuing Subcommand Input Strings	40
File Name Formats	41

Fully-Qualified Pathnames	44
Storage Space Requirements for Transferred Files.	44
File Transfer Methods	45
File List Formats	46
Transferring Files Using a Web Browser	50
ACCT	51
APPEND	52
ASCII	52
BINARY	53
CD or CWD	54
CDUP	57
CLOSE.	57
CMS.	57
DEBUG	58
DELETE	58
DELIMIT	59
DIR	59
EBCDIC	61
EUCKANJI	61
GET	61
HANGEUL	62
HELP	63
IBMKANJI.	63
JIS78KJ	64
JIS83KJ	64
KSC5601	65
LCD	65
LOCSITE	66
LOCSTAT	66
LPWD	67
LS	67
MDELETE	69
MGET	69
MKDIR	70
MODE	71
MPUT	71
NETRC.	72
NOOP	73
OPEN	73
PASS	74
PASSIVE	75
PUT	75
PWD	76
QUIT	77
QUOTE	77
RENAME	78
RMDIR	79
SENDPORT	79
SENDSITE	80
SITE.	80
SIZE.	83
SJISKANJI	83
STATUS	83
STRUCT	84
SUNIQUE.	84
SYSTEM	86

TCHINESE	86
TYPE	87
USER	88
Using FTP Within an EXEC	89
Providing FTP Subcommand Input.	89
Managing FTP Command Output	89
Examples	90
FTP Return Codes	92
Examples	93
FTP Reply Codes	94
Internal Error Codes	95
Using FTP with RACF	96
 Chapter 3. Transferring Files Using TFTP	97
TFTP Command	97
Creating Translation Tables	98
TFTP Subcommands	98
GET	98
HELP	99
LOCSTAT	99
MAXPKT	99
MODE	100
OPEN.	100
PUT	101
QUIT	101
REXMIT	101
TRACE	102
 Chapter 4. Sending and Receiving Electronic Mail	105
NOTE and SENDFILE Commands	105
Electronic Mail Gateway	105
Note and File Delivery.	106
SMTPQUEUE Command	106
Undelivered Notes	106
Nondelivery Note	107
Unknown Recipient Note	107
Using OfficeVision Without OfficeVision Extended Mail Installed	108
Specifying TCP/IP Recipients	108
Example of Sending a Note Copy to SMTP	109
Note Delivery	110
Using IMAP to Access Electronic Mail	110
Using the SMSG Interface to SMTP.	110
General User SMSG Commands	110
Receiving Electronic Mail on VM	113
 Chapter 5. Logging On to a Foreign Host	115
TELNET Command	115
TELNET Function Keys	116
In Transparent (TN3270) Mode	116
In Line Mode	116
TELNET Subcommands	117
AO	117
AYT	117
BRK	118
HELP	118
IP	118

PA1	119
QUIT	119
SYNCH	119
Sending ASCII Control Characters to a Host in Line Mode	120
Chapter 6. Monitoring the TCP/IP Network	121
NETSTAT—Displaying Local Host Information	121
NETSTAT Command Address Interpretation	121
NETSTAT Command	122
Examples	135
RPCINFO Command	155
TRACERTE Command	156
PING Command	159
Chapter 7. Authenticating Network Users with Kerberos	163
Understanding Kerberos Name Structures	163
Kerberos Commands	163
KINIT Command	164
KLIST Command	165
KDESTROY Command	166
KPASSWORD Command	167
Chapter 8. Using the Network File System Commands	169
VM's NFS Server Support	169
NFS Client Commands	170
MOUNT Command	170
MOUNTPW Command	180
UMOUNT Command	181
Diagnosing Mount Problems	182
Errors Using Mounted Systems	183
Notes for BFS Files and Directories	185
Notes for SFS Files and Directories	186
Notes for Minidisk Files	188
SMSG Interface to VMNFS	190
SMSG DETACH Command	190
SMSG ERROR Command	191
SMSG QUERY Command	192
SMSG REFRESH user_id.vaddr Command	195
Name Translation File	196
Special File Names for VM NFS	196
Special File Name Considerations	197
NFS Client Problems	197
Deleting CMS Record-Length Fields	198
Using NFS with RACF	198
VM NFS Server Link Support	198
SFS and Minidisk Links	198
BFS Links	198
Chapter 9. Using the Remote Execution Protocol	201
REXEC Command	201
The NETRC DATA File	203
Anonymous Remote Command Execution	203
Command Execution Using Your Own Virtual Machine	203
Notes and Restrictions	204
Using RSH commands with REXECD	205

Chapter 10. Using Remote Printing	207
LPRSET Command	207
LPR Command	211
Controlling LPSERVE from other Systems	223
LPQ Command	224
LPRM Command	226
 Chapter 11. Using GDDMXD/VM with the X Window System	229
Overview of GDDMXD/VM	229
GDDM Application Limitations	229
GDDM Display Limitations	229
z/VM Considerations	230
GDDMXD Command	230
Configuring the GDDMXD/VM Environment	231
Identifying the Target Display	231
GLOBALV Command	231
User-Specified Commands	232
Resizing the GDDMXD Graphics Window	232
gddmx*Geometry: Command	233
gddmx*GColornn: Command	234
gddmx*GMCPnn: Command	235
gddmx*ZWL: Command	235
gddmx*XSync: Command	236
gddmx*CMap: Command	236
gddmx*HostRast: Command	237
gddmx*XCIconn: Command	237
gddmx*Compr: Command	238
gddmx*PDTrace: Command	238
gddmx*ANFontn Command	239
Remapping the Enter and Newline Keys	239
gddmx*Enter: Command	239
gddmx*NewLine: Command	240
GDDMXD/VM Keyboard Functions	240
GDDMXD/VM to X Window System Keyboard Functions	241
APL2 Character Set Keyboard	241
Setting Up GDXXAPLCS MAP	242
Collecting Problem Determination Information	242
GDDMXD/VM—X Window System Server Environment	242
Obtaining the GDDM Trace	243
Obtaining the GDDMXD Problem Determination Trace	243
GDDMXD/VM Problem Determination Examples	243
Demonstration Programs	244
GXDEMO1	244
GXDEMO2	244
GXDEMO3	244
GXDEMO4	244
GXDEMO4A	244
GXDEMO5	244
GXDEMO6	245
 Chapter 12. Managing TCP/IP Network Resources with SNMP	247
Sample Command Lists	247
SNMP/NetView Overview	248
SNMP Commands	249
SNMP Commands Overview	249
Return Codes	249

SNMP GET Command	250
SNMP GETNEXT Command	252
SNMP SET Command.	254
SNMP TRAPSON Command	255
SNMP TRAPSOFF Command	258
SNMP MIBVNAME Command	259
SNMP PING Command	260
Major and Minor Error Codes and SNMP Value Types	261
gethostbyname()	262
IBM 3172 Enterprise-Specific MIB Variables.	262
 Chapter 13. Using the Network Database System (NDB)	265
The Client Machine	266
Components of the Client Machine	266
The Server Machine	267
Components of the Portmap Manager Server	267
Components of the NDB Server	268
Connecting to a Database	269
NDBCLNT Command	270
Format of Output Displayed on the Client.	271
SQL Commands from a Remote Machine	272
SQL Commands from an Application	272
Security	274
Limitations	274
Additional information on the NDB Client code	275
 Chapter 14. Using the Domain Name System	277
Overview of the Domain Name System	277
Domain Names	277
Domain Name Servers	278
Resolvers	279
Resource Records	280
NSLOOKUP—Querying Name Servers	283
NSLOOKUP Internal State Information.	283
NSLOOKUP Commands	283
NSLOOKUP Command Line Query	284
NSLOOKUP Interactive Session Queries	285
NSLOOKUP Examples	291
DIG—Querying Name Servers.	294
DIG Internal State Information	294
DIG Command	294
DiG Examples.	300
CMSRESOL—Resolver and Name Server	305
RESOLV Command—Interface to the CMSRESOL MODULE	305
CMSRESOL Command—Interface to the Resolver	306
Types of Queries.	307
The Standard Query	307
The Inverse Query	307
Querying the in-addr.arpa Domain	308
The Database Query and Update	308
Querying Outside Your Domain	309
 Chapter 15. Using Translation Tables	311
Character Sets and Code Pages	311
TCP/IP Translation Table Files.	312
Translation Table Search Order	312

Special Telnet Requirements	313
IBM-Supplied Translation Tables	314
Customizing SBCS Translation Tables	316
Syntax Rules for SBCS Translation Tables	317
Customizing DBCS Translation Tables	317
DBCS Translation Table	318
Syntax Rules for DBCS Translation Tables	318
Sample DBCS Translation Tables	318
Converting Translation Tables to Binary	320
CONVXLAT Command	321
Appendix A. Specifying Files and Data Sets.	323
AIX Files.	323
OS/390 Data Sets	323
Sequential Data Sets	324
Partitioned Data Sets	324
DOS, OS/2, and Windows 98 Files	325
AS/400 Operating System	326
Appendix B. Using the NETRC DATA File.	327
Using The NETRC DATA File with FTP	327
Using The NETRC DATA File with REXEC	327
Using the NETRC DATA File with the OPEN MOUNT CMS Command	328
Appendix C. Mapping Values for the APL2 Character Set	329
Appendix D. Using DBCS with FTP and Mail	335
Using DBCS with FTP.	335
DBCS Translation Tables.	335
DBCS Subcommands	335
Defining FTP with Kanji Support	338
Using FTP with Kanji Support	338
Using DBCS with Mail.	339
SMTP Server DBCS Support	339
SMTP Protocol for 8-bit characters	339
Conversion of DBCS Mail	339
Conversion of DBCS Files	340
Appendix E. Management Information Base Objects	341
Structure and Identification of Management Information (SMI)	341
Names	342
The Management Subtree	342
MIB/Network Elements	344
System Group.	345
Interfaces Group.	346
Address Translation Group	350
IP Group.	351
IP Address Translation Table	356
ICMP Group	357
TCP Group.	359
UDP Group.	361
IBM 3172 Enterprise-Specific MIB Variables.	362
ibm3172SystemTable	362
ibm3172ifTrapTable.	363
ibm3172ifChanCounters Table	364
ibm3172ifLANCounters Table	365

ibm3172ifBlkCounters Table.	366
ibm3172ifDbkCounters Table	367
ibm3172ifDeviceTable	368
Appendix F. IBM 3172 Attribute Index	369
Appendix G. SNMP Generic TRAP Types	371
Appendix H. Related Protocol Specifications	373
Appendix I. Abbreviations and Acronyms.	377
Notices	381
Programming Interface Information	382
Third Party Copyright Information.	383
Trademarks.	387
Glossary	389
Bibliography	407
Where to Get z/VM Books	407
z/VM Base Library	407
Overview	407
Installation, Migration, and Service	407
Planning and Administration.	407
Customization and Tuning	407
Operation	407
Application Programming.	407
End Use	408
System Diagnosis	408
Books for z/VM Optional Features	408
Data Facility Storage Management Subsystem for VM	408
Directory Maintenance Facility.	408
Performance Toolkit for VM	409
Resource Access Control Facility.	409
Other TCP/IP Related Publications	409
Index	411

About This Book

This book describes the functions of and explains how to use the applications available in TCP/IP running on the IBM® z/VM® V5.2 operating system (TCP/IP level 520). 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 book 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 *TCP/IP Planning and Customization* and *TCP/IP Programmer's Reference*. For information about error messages that may appear while using TCP/IP, see *TCP/IP Messages and Codes*.

Who Should Read This Book

This book is intended for an end-user and describes how to use level 520 after it has been installed and customized on your network. You should read this book when you want to use the applications that are available in level 520.

What You Should Know before Reading This Book

Before using this book, 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.

How to Use This Book

Use this book to understand the commands that are used to exploit the networking protocols.

How to Read Syntax Diagrams

This book uses diagrams to show the syntax of external interfaces and statements. Also, this book uses a special notation to show variable, optional, or alternative content in examples of messages and responses.

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 following examples.

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.	►—KEYWORD—◄◄														
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.															
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.	►—KEYWOrd—◄◄														
In this example, you can specify KEYWO, KEYWOR, or KEYWORD.															
Symbols:															
You must specify these symbols exactly as they appear in the syntax diagram.	<table><tr><td>*</td><td>Asterisk</td></tr><tr><td>:</td><td>Colon</td></tr><tr><td>,</td><td>Comma</td></tr><tr><td>=</td><td>Equal Sign</td></tr><tr><td>-</td><td>Hyphen</td></tr><tr><td>()</td><td>Parentheses</td></tr><tr><td>.</td><td>Period</td></tr></table>	*	Asterisk	:	Colon	,	Comma	=	Equal Sign	-	Hyphen	()	Parentheses	.	Period
*	Asterisk														
:	Colon														
,	Comma														
=	Equal Sign														
-	Hyphen														
()	Parentheses														
.	Period														
Variables:															
A variable appears in highlighted lowercase, usually italics.	►—KEYWOrd— <i>var_name</i> —◄◄														
In this example, <i>var_name</i> represents a variable that you must specify following KEYWORD.															
Repetitions:															
An arrow returning to the left means that the item can be repeated.	►—repeat—◄◄ 														
A character within the arrow means that you must separate each repetition of the item with that character.	►—repeat—◄◄ 														
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.	►—repeat—◄◄ 														
Syntax notes may also be used to explain other special aspects of the syntax.	Notes: 1 Specify <i>repeat</i> up to 5 times.														

Syntax Diagram Convention

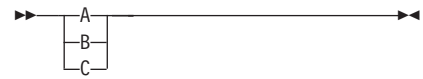
Example

Required Item or Choice:

When an item is on the line, it is required. In this example, you must specify A.

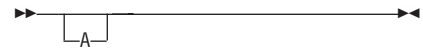


When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.



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.

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.



A Fragment:



In this example, the fragment is named "A Fragment."

Message and Response Notation

This book may include examples of messages or responses. Although most messages and responses are shown exactly as they would appear, some content may 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 items that may be displayed.

{ } Braces enclose alternative items, only one of which will be displayed.

| The vertical bar separates items within brackets or braces.

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

How the Term “internet” Is Used in This Book

In this book, 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 Book

In this book, 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.

What This Book Contains

The following section briefly describes the format and organization of this book. This book describes all applications available with level 520; however, your organization may install and use only some of these functions.

Chapter 1, “Introducing Computer Networks and Protocols,” on page 1, introduces the concepts of computer networks, the internet environment, TCP/IP protocols and functions, routing, and internet addressing. This chapter is optional for users who are familiar with computer networks and protocols.

Chapter 2, “Transferring Files Using FTP,” on page 27, describes how to use the File Transfer Protocol (FTP) command and its subcommands to transfer files from one host to another host.

Chapter 3, “Transferring Files Using TFTP,” on page 97, describes how to use the Trivial File Transfer Protocol (TFTP) command and the subcommands that are associated with the TFTP command.

Chapter 4, “Sending and Receiving Electronic Mail,” on page 105, describes how to send electronic mail using the NOTE command, the SENDFILE command, and the PROFS® interface.

Chapter 5, “Logging On to a Foreign Host,” on page 115, describes how to log on to a foreign host using the various types of terminal emulation that are associated with the Telnet protocol.

Chapter 6, “Monitoring the TCP/IP Network,” on page 121, describes how to obtain status information from the network using the NETSTAT command, RPCINFO command, and PING command.

Chapter 7, “Authenticating Network Users with Kerberos,” on page 163, describes how to use the authentication services of Kerberos.

Chapter 8, “Using the Network File System Commands,” on page 169, describes how to use the Network File System commands.

Chapter 9, “Using the Remote Execution Protocol,” on page 201, describes how to use the REXEC command.

Chapter 10, “Using Remote Printing,” on page 207, describes the line printer daemon (LPD) that provides support for remote printing.

Chapter 11, “Using GDDMXD/VM with the X Window System,” on page 229, describes GDDMXD/VM and the GDDMXD command. This chapter also describes how to use GDDMXD/VM user-specified options and keyboard functions.

Chapter 12, “Managing TCP/IP Network Resources with SNMP,” on page 247, describes the Simple Network Management Protocol (SNMP).

Chapter 13, “Using the Network Database System (NDB),” on page 265, describes the Network Database system (NDB) used for relational database systems in a TCP/IP environment.

Chapter 14, “Using the Domain Name System,” on page 277, describes Domain Name Systems and how to query domain name servers.

Chapter 15, “Using Translation Tables,” on page 311, describes how to customize the translation tables supplied with TCP/IP.

Appendix A, “Specifying Files and Data Sets,” on page 323, describes the acceptable file formats for the AIX®, MVS™, and OS/2® operating systems, and AS/400's® operating system. This appendix also provides examples of each format.

Appendix B, “Using the NETRC DATA File,” on page 327, describes the NETRC DATA File and how it can be used with various applications.

Appendix C, “Mapping Values for the APL2 Character Set,” on page 329, describes the GDXAPLCS MAP file. This appendix also lists GDDMXD/VM's default mapping values for the APL2® character set.

Appendix D, “Using DBCS with FTP and Mail,” on page 335, describes how to use National Language Support to transfer Kanji files using FTP and SMTP.

Appendix E, “Management Information Base Objects,” on page 341, describes the Management Information Base (MIB) objects that are supported by the VM agent.

Appendix F, “IBM 3172 Attribute Index,” on page 369, lists MIB variables and corresponding attributes.

Appendix G, “SNMP Generic TRAP Types,” on page 371, describes the TRAP messages associated with SNMP.

Appendix H, “Related Protocol Specifications,” on page 373, lists the related protocol specifications concerning level 520.

Appendix I, “Abbreviations and Acronyms,” on page 377, lists the abbreviations and acronyms that are used throughout this book.

This book also includes a glossary, a bibliography, and an index.

Where to Find More Information

Appendix I, “Abbreviations and Acronyms,” on page 377, lists the abbreviations and acronyms that are used throughout this book.

The “Glossary” on page 389, defines terms that are used throughout this book associated with TCP/IP communication in an internet environment.

For more information about related publications, see “Bibliography” on page 407.

Links to Other Online Books

If you are viewing the Adobe Portable Document Format (PDF) version of this book, it may contain links to other books. A link to another book is based on the name of the requested PDF file. The name of the PDF file for an IBM book is unique and identifies both the book and the edition. The book links provided in this book are for the editions (PDF names) that were current when the PDF file for this book was generated. However, newer editions of some books (with different PDF names) may exist. A link from this book to another book works only when a PDF file with the requested name resides in the same directory as this book.

How to Send Your Comments to IBM

IBM welcomes your comments. You can use any of the following methods:

- Complete and mail the Readers’ Comments form (if one is provided at the back of this book) or send your comments to the following address:

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, New York 12601-5400
U.S.A.

- Send your comments by FAX:
 - United States and Canada: 1-845-432-9405
 - Other Countries: +1 845 432 9405
- Send your comments by electronic mail to one of the following addresses:
 - Internet: mhvrcfs@us.ibm.com
 - IBMLink™ (US customers only): IBMUSM10(MHVRCFS)

Be sure to include the following in your comment or note:

- Title and complete publication number of the book
- Page number, section title, or topic you are commenting on

If you would like a reply, be sure to also include your name, postal or email address, telephone number, or FAX number.

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

Summary of Changes

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change (in that edition only). Some product changes identified in this summary may be provided through z/VM service by program temporary fixes (PTFs) for authorized program analysis reports (APARs).

SC24-6127-01, z/VM Version 5 Release 2

This edition supports the general availability of z/VM Version 5 Release 2 (z/VM V5R2).

Changes to the NETSTAT Command

A new CONFIG option has been added to allow NETSTAT to query stack configuration information.

SC24-6127-00, z/VM Version 5 Release 1

This edition supports the general availability of z/VM Version 5 Release 1 (z/VM V5R1).

Internet Protocol Version 6 (IPv6) Support

Changes have been made for enhanced IPv6 support. The TCP/IP for z/VM stack can be configured for IPv6 networks connected through OSA-Express operating in QDIO mode. The stack can be configured to provide static routing of IPv6 packets and to send IPv6 Router Advertisements. The native TCP/IP for z/VM applications that have been enhanced to support IPv6 are TRACERTE and PING.

Enabled for LibraryCenter Advanced Search

This book has been enabled for the following z/VM LibraryCenter advanced searches: commands.

SC24-6020-02, z/VM Version 4 Release 4

This edition supports the general availability of z/VM Version 4 Release 4 (z/VM V4R4).

Various technical as well as editorial corrections have been made to this book. In addition, some restructuring to the command syntax and layouts has occurred in order to provide HELP information for the TCP/IP user commands.

TCP/IP Stack Security

Logon By support has been added to REXEC Command.

Virtual Switch Support

Information has been added to NETSTAT Command to support z/VM Virtual Switches and VM LAN segments.

SC24-6020-01, z/VM Version 4 Release 3

This edition supports the general availability of z/VM Version 4 Release 3 (z/VM V4R3).

PASV FTP client support

A new FTP PASSIVE subcommand has been added so that z/VM FTP clients outside of a firewall can transfer files to and from an FTP server that is inside of a firewall.

TCP/IP Stack Vulnerability Reduction

Function has been added to improve the performance and reliability of the TCP/IP stack by preventing more Denial-of-Service (DoS) attacks. These attacks include:

- Kiss-of-Death (KOD) — an IGMP based attack that depletes the stack's large envelopes
- KOX — a version of the KOD attack that also has source IP address spoofing
- Stream — an attack in which TCP packets are sent to the stack with no header flags set
- R4P3D — an augmented version of the Stream attack
- Blat — a version of the Land attack that also has the URG flag turned on in the TCP header and has the ability to incrementally spoof the source IP address
- SynFlood — an attack in which the initiator floods the TCP/IP stack with SYN packets that have spoofed source IP addresses, resulting in the server never receiving the final ACKs needed to complete the three-way handshake in the connection process.

The Smurf DoS attack has also been updated to address three variants of the attack. Smurf is a DoS attack in which an ICMP Echo Request is sent to a broadcast or multicast address. The three variants are:

- Smurf-IC — where "IC" denotes that incoming packets are using the TCP/IP stack to launch an attack
- Smurf-OB — where "OB" denotes that an outbound ICMP Echo Request matched the description of a Smurf attack
- Smurf-RP — where "RP" denotes that ICMP Echo Reply packets being received by the stack do not match any Echo Requests that were sent.

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 Token-Ring, Ethernet, PC Network, Fiber Distribution Data Interface (FDDI), and X.25. 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.

Introduction

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 14.

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 Token-Ring or 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, visit the following Web site:

http://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 the following Web site:

<http://www.redbooks.ibm.com>

The TCP/IP for z/VM IPv6 support improves the guest LAN support for the OSA-Express simulation in QDIO mode. Virtual machines (z/VM and other guest operating systems) in the guest LAN environment are able to define and to use simulated OSA-Express devices that support both the IPv4 and IPv6 protocols. The IP Assist Simulation for QDIO-based 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 4.) 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 FTP, and 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*, SC24-6126
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6105
- *C/C++ for z/VM: Run-Time Library Reference*, SC09-7624

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.

Copies of these RFCs are available at the following Web site:

<http://www.ietf.org/html.charters/ipv6-charter.html>

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 1 on page 5 shows the functional groups and their related protocols and functions.

Table 1. Functional Groups

Group	Protocols and Functions	Page
Link (physical) layer	Token-Ring Ethernet X.25 Others	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	6
Transport Layer	Transmission Control Protocol (TCP) User Datagram Protocol (UDP)	8
Application Layer	Telnet File Transfer Protocol (FTP) Trivial File Transfer Protocol (TFTP) Internet Message Access Protocol (IMAP) Simple Mail Transfer Protocol (SMTP) Domain Name System (DNS) Simple Network Management Protocol (SNMP) Kerberos Authentication System Remote Printing (LPR and LPD) RouteD X Window System X Toolkit GDDMXD Remote Procedure Call (RPC) Network File System (NFS) Remote Execution Protocol (REXEC) Bootstrap Protocol (BOOTP) Dynamic Host Configuration Protocol (DHCP) Network Database System (NDB) Socket Interfaces Secure Socket Layer (SSL)	9 - 13

Figure 1 on page 6 shows the relationship of these protocols and functions within the TCP/IP layered architecture for VM.

TCP/IP Protocols and Functions

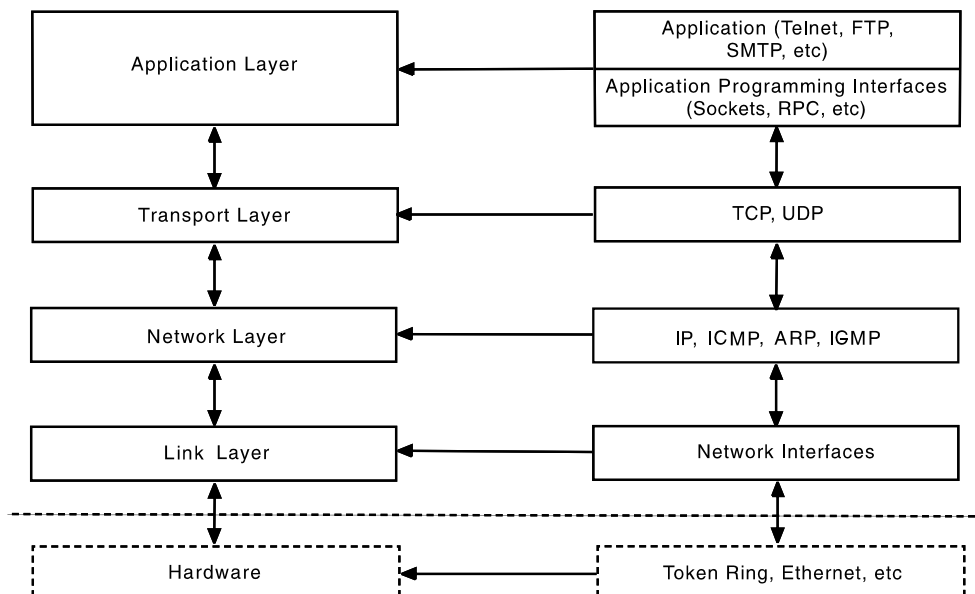


Figure 1. The TCP/IP Layered Architecture

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.

X.25

You can use an X.25 network to establish a TCP/IP connection between two hosts. X.25, recommended as a communication interface standard by the International Telegraph and Telephone Consultative Committee (CCITT), defines the interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE). A DTE is a computer or workstation connected to a network. A DCE is the equipment at the point of the connection to the network, such as a modem.

For more information about using TCP/IP over an X.25 network, see Request For Comment (RFC) 877.

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 H, "Related Protocol Specifications."

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

TCP/IP Protocols and Functions

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.

Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol (TFTP) is designed only to read and write files to and from a foreign host. TFTP is built on the services of UDP in the transport layer. TFTP allows you to limit drive and directory access.

TCP/IP Protocols and Functions

TFTP, like FTP, can transfer files as either ASCII characters or as binary data. However, unlike FTP, TFTP cannot be used to list or change directories at a foreign host, and it has no provisions for user authentication.

For more information about TFTP, see RFC 783.

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.

Kerberos Authentication System

The Kerberos Authentication System provides additional security by allowing authorization checking at the user level rather than at the node level. This system allows client and server pairs to verify that the partner is authorized to participate in the function being performed.

For additional publications about Kerberos, see Bibliography.

Remote Printing (LPR and LPD)

TCP/IP provides both client and server 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.

RouteD

RouteD uses the Routing Information Protocol (RIP) to dynamically create and maintain network routing tables. RIP arranges to have gateways and routers periodically broadcast their routing tables to neighbors. Using this information, a RouteD server can update a host's routing tables. For example, RouteD 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.

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.

Bootstrap Protocol (BOOTP)

The Bootstrap Protocol (BOOTP) allows a diskless client machine to discover its own IP address, the address of a server host, and the name of a file to be loaded into memory and executed.

For more information about BOOTP, see RFC 0951 and *TCP/IP Planning and Customization*.

Dynamic Host Configuration Protocol (DHCP)

The Dynamic Host Configuration Protocol (DHCP) provides a framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on the Bootstrap Protocol (BOOTP), adding the capability of automatic allocation of reusable network addresses and additional configuration options. DHCP captures the behavior of BOOTP relay agents; DHCP participants can interoperate with BOOTP participants.

For more information about DHCP, see RFC 2131 and *TCP/IP Planning and Customization*.

Network Database System (NDB)

The Network Database System (NDB) is used to access relational database systems in a TCP/IP-based internet environment. NDB uses the RPC protocol to allow interoperability among a variety of workstation users and the mainframe

database management system, DB2 Server for VM NDB is used for data conversion, security, I/O buffer management, and transaction management.

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 *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 SSL, see *TCP/IP Planning and Customization*.

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

IPv4 Addressing

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.

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.

[illegible]

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.

		1												2												3											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1						
1	0	Network										Local Address																									

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.

0 1 2	3 4 5 6 7 8 9 0 ¹ 1 2 3 4 5 6 7 8 9 0 ² 1 2 3	4 5 6 7 8 9 0 ³ 1
1 1 0	Network	Local Address

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.

0 1 2 3	4 5 6 7 8 9 0 ¹ 1 2 3 4 5 6 7 8 9 0 ² 1 2 3 4 5 6 7 8 9 0 ³ 1
1 1 1 0	Multicast Address

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.///.///.///*
Class B *nnn.nnn.///.///*
Class C *nnn.nnn.nnn.///*

where:

nnn Represents part or all of a network number.
/// 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:

<i>network_number</i>	Is the network portion of the internet address.
<i>subnet_number</i>	Is a field of a constant width for a given network.
<i>host_number</i>	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 represents a class B address with a 6-bit wide subnet field.

0 1	2 3 4 5 6 7 8 9 0 1 2 3 4 5	6 7 8 9 0 1	2 3 4 5 6 7 8 9 0 1
1 0	Network	Subnet	Host

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 addressability 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, where the x's are the hexadecimal values of the eight 16-bit pieces of the address.

Examples:

FE80:0000:0000:0001:0800:23e7: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: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
0:0:0:0:0:0: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:

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

Figure 7. IPv6 Address Format

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).

TCP/IP Protocols and Functions

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::CD30:0:0:0/60

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 2. 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:

n bits	128-n bits
network prefix	interface ID

Figure 8. Unicast Address Format

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:

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

Figure 9. Global Unicast Address Format

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.

TCP/IP Protocols and Functions

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:

10 bits	54 bits	64 bits
1111111010	0	interface ID

Figure 10. Link-local address format

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.

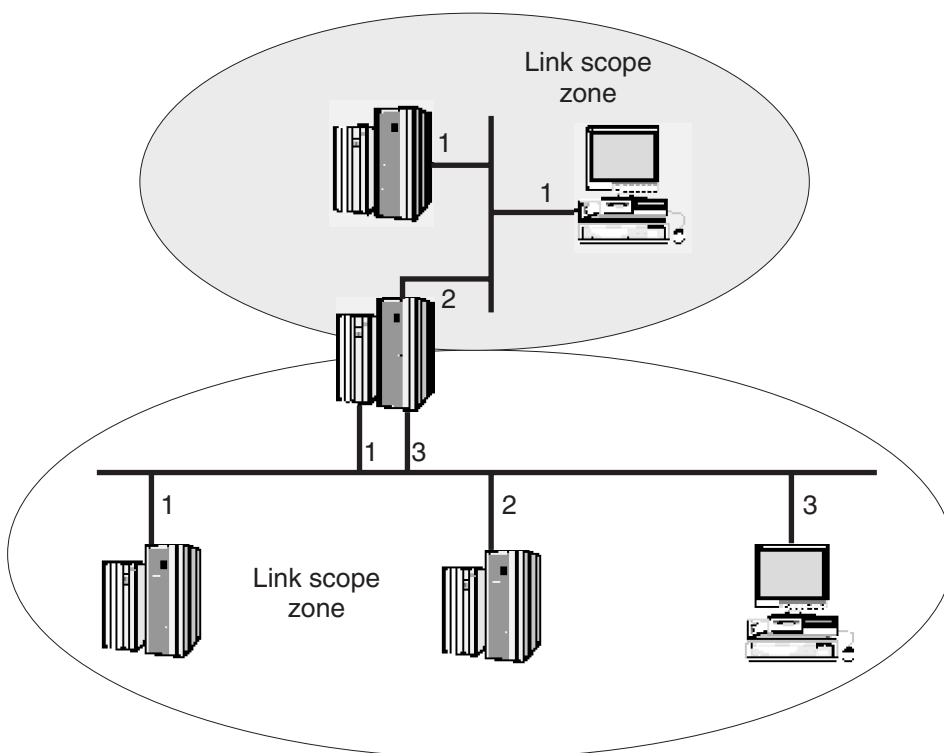


Figure 11. Link-local scope zones

Figure 11 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:

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

Figure 12. Site-local address format

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.

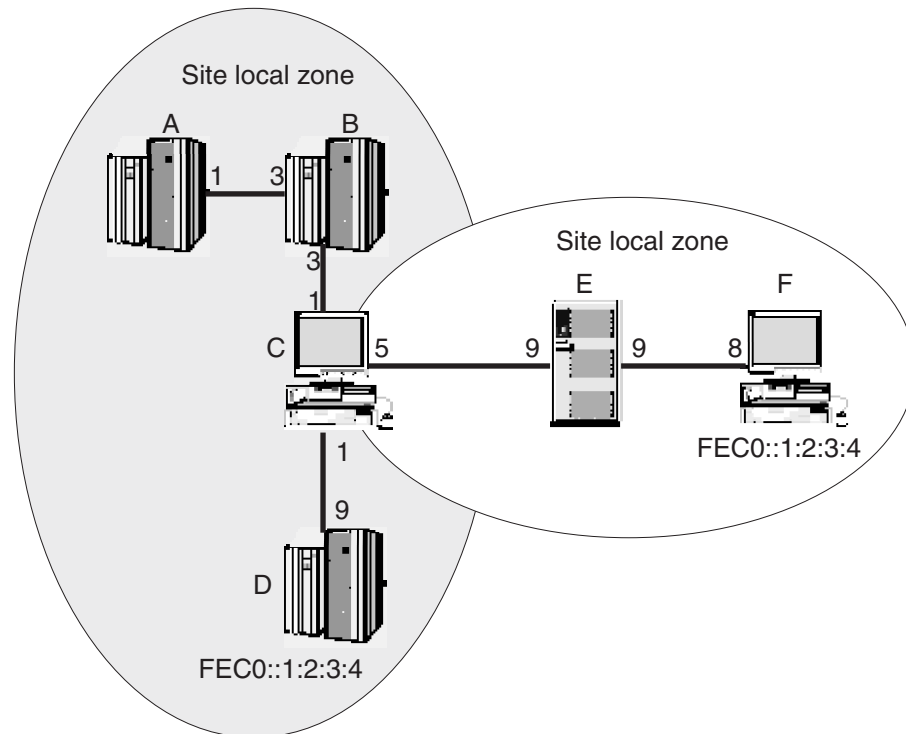


Figure 13. Site-local scope zones

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 13 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

TCP/IP Protocols and Functions

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 13 on page 21, 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 0:0:0:0:0:0:1 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 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:

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

Figure 14. IPv4-mapped IPv6 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:

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

Figure 15. IPv4-compatible IPv6 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:

8	4	4	112 bits
11111111	flags	scope	group ID

Figure 16. Multicast address format

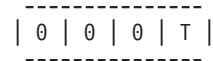


Figure 17. Flags in multicast address

- 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 3. Multicast scope field values

Value	Scope
0	Reserved
1	Interface-local scope (same node)

Table 3. Multicast scope field values (continued)

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.

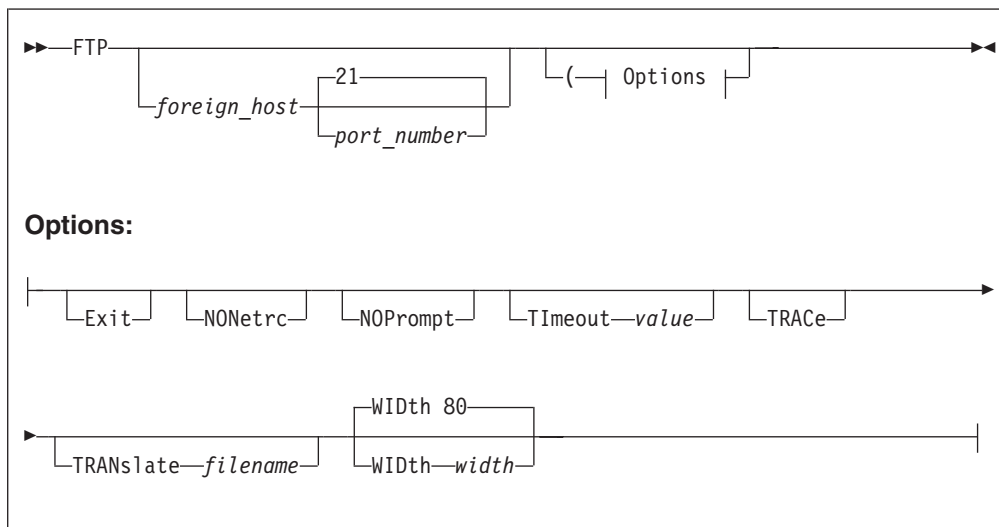
When FTP commands are used with a non-z/VM foreign host, you may need to consult documentation specific to that host and its operating system for information about file naming and supported FTP commands and parameters.

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

Table 4. 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	

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 5 on page 38 can be used to transfer files and interact with the remote FTP server.

Operands

foreign_host

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

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.

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 92.

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.

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.

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 311.

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 occurs, 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

Transferring Files Using FTP

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 40.

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 520
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 327.

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.

Notes:

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

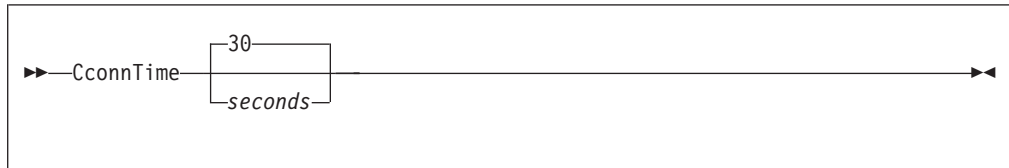
Within FTP DATA, blanks and record boundaries are used to delimit tokens. All characters to the right of (and including) a semicolon are treated as comments.

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.



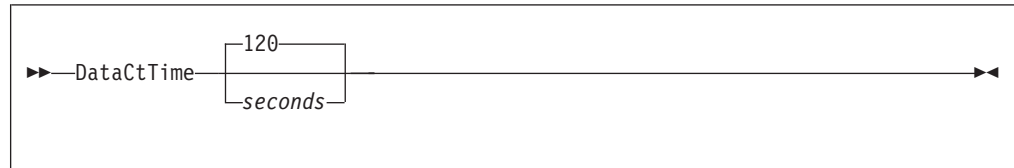
Operands

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.



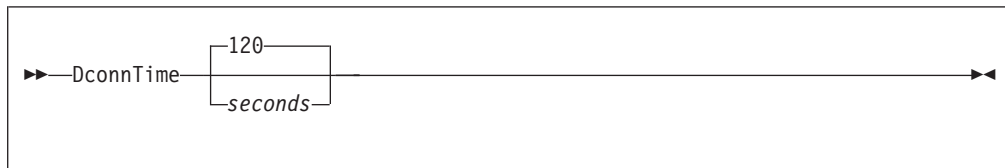
Operands

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.



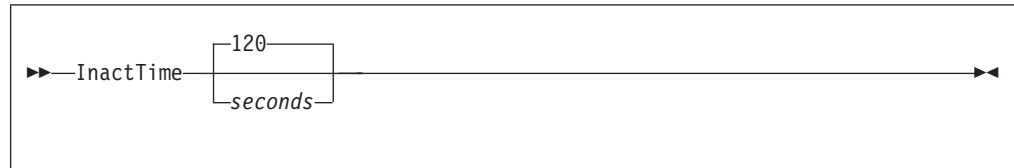
Operands

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.

INACTTIME Statement

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



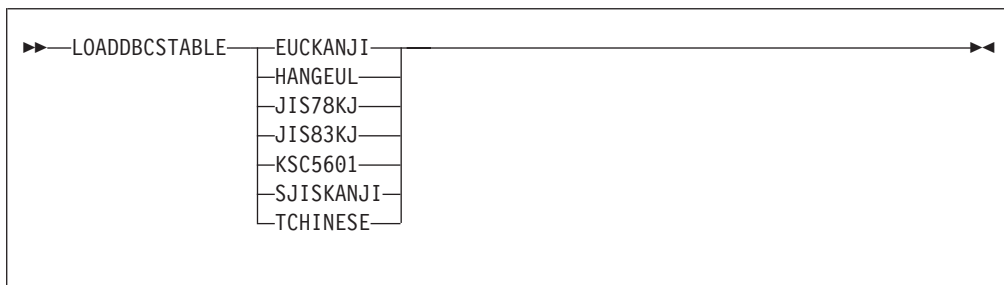
Operands

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.

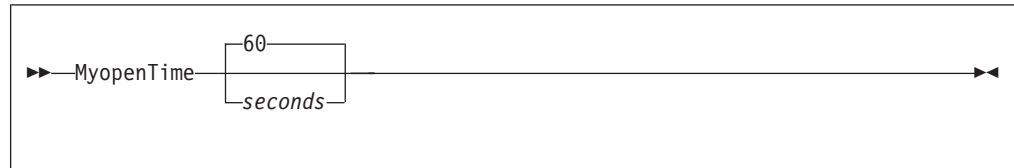
If the LOADDBCSTABLE parameter is not specified, specified incorrectly, or if FTP DATA is not accessible, then no DBCS translate tables will be loaded, and the corresponding FTP client DBCS transfer types will be unavailable.

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.

FTP Subcommands

The FTP subcommands listed in Table 5 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 5 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 5. FTP Subcommands

Subcommand	Minimum Abbreviation	Description	Page
ACCT	AC	Sends host-dependent account information.	51
APPEND	AP	Appends a file on your local host to a file on the foreign host.	52
ASCII	AS	Sets the transfer type to ASCII.	52
BINARY	B	Sets the transfer type to IMAGE.	53
CD	CD	Changes the working directory. CWD is a synonym for CD.	54
CDUP	CDU	Changes the working directory to the parent directory on the foreign host. CD is a synonym for CDUP.	57
CLOSE	CL	Disconnects from the foreign host.	57
CMS	CM	Passes a CMS command.	57
CWD	CW	Changes the working directory. CD is a synonym for CWD.	54
DEBUG	DEB	Toggles internal debug options.	58
DELETE	DELE	Deletes a single file on the foreign host.	58
DELIMIT	DELI	Displays the delimiter character between the file name and the file type.	59
DIR	DI	Lists the directory entries for files on the foreign host. LIST is a synonym for DIR.	59
EBCDIC	EB	Sets the transfer type to EBCDIC.	61
EUCKANJI	EU	Sets the transfer type to EUCKANJI.	61
GET	G	Copies a file from the foreign host to your local host. RETRIEVE is a synonym for GET.	61
HANGEUL	HA	Sets the transfer type to HANGEUL.	62
HELP	H	Displays help information for FTP.	63
?	?	Provides information to use FTP.	63
IBMKANJI	I	Sets the transfer type to IBMKANJI.	63
JIS78KJ	JIS7	Sets the transfer type to JIS78KJ.	64
JIS83KJ	JIS8	Sets the transfer type to JIS83KJ.	64
KSC5601	K	Sets the transfer type to KSC5601.	65

Table 5. FTP Subcommands (continued)

Subcommand	Minimum Abbreviation	Description	Page
LCD	LC	Changes the working directory on the local host.	65
LOCSITE	LOCSI	Changes local site information.	66
LOCSTAT	LOCST	Displays FTP status information for the local host.	66
LPWD	LP	Displays the name of the active working directory on the local host.	67
LS	LS	Lists the names of files on the foreign host. NLST is a synonym for LS.	67
MDELETE	MD	Deletes multiple files on the foreign host.	69
MGET	MG	Copies multiple files from the foreign host to your local host.	69
MKDIR	MK	Creates a new directory on the foreign host.	70
MODE	MO	Specifies the mode or data format of the transfer.	71
MPUT	MP	Copies multiple files on your local host to the foreign host.	71
NETRC	NE	Enables or disables automatic logon capability through use of a NETRC DATA file.	72
NOOP	NO	Checks whether the foreign host is still responding.	73
OPEN	O	Opens a connection to a foreign host. CONNECT is a synonym for OPEN.	73
PASS	PA	Supplies a password to the foreign host.	74
PASSIVE	PASSI	Controls whether the client or server establishes connections for data transfers.	75
PUT	PU	Copies a file on your local host to the foreign host.	75
PWD	PW	Displays the name of the active working directory on the foreign host.	76
QUIT	QUI	Leaves the FTP command environment.	77
QUOTE	QUO	Sends an uninterpreted string of data.	77
RENAME	REN	Renames a file on the foreign host.	78
RMDIR	RM	Remove a directory from the foreign host.	79
SENDPORT	SENDP	Enables or disables automatic transmission of the FTP server PORT subcommand. TOGLPORT is a synonym for SENDPORT.	79
SENDSITE	SENDS	Enables or disables automatic transmission of the SITE subcommand.	80
SITE	SI	Sends information to the foreign host using site-specific commands.	80
SIZE	SIZE	Retrieves the transfer size (in bytes) for a foreign host file.	83

Table 5. FTP Subcommands (continued)

Subcommand	Minimum Abbreviation	Description	Page
SJISKANJI	SJ	Sets the transfer type to SJISKANJI.	83
STATUS	STA	Displays status information for the foreign host.	83
STRUCT	STR	Sets the file transfer structure.	84
SUNIQUE	SU	Toggles the storage methods.	84
SYSTEM	SY	Displays the name of the foreign host's operating system.	86
TCHINESE	TC	Sets the transfer type to TCHINESE.	86
TYPE	TY	Specifies the transfer type.	87
USER	U	Identifies you to a foreign host. LOGIN is a synonym for USER.	88

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.worldfiles
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.
```

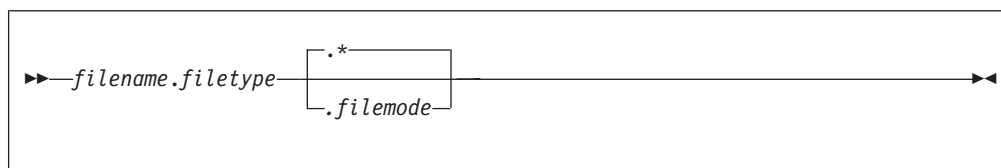
File Name Formats

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:

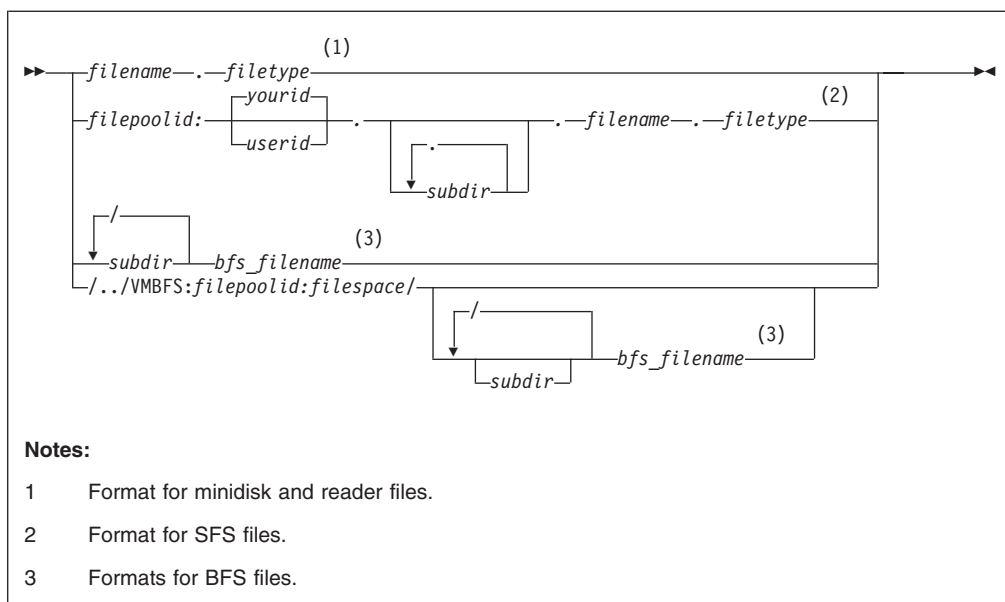


Notes:

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:

FTP Subcommands



To reference minidisk or reader files, the appropriate working directory must be established on the foreign VM host. Fully-qualified SFS and BFS 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.

Notes:

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.
2. Pattern matching cannot be used for BFS files and directories.

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 323 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.

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 (') quotes.

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

FTP Subcommands

If quotes are encountered that enclose a subcommand string which does *not* contain embedded blanks, the z/VM FTP client assumes the supplied quotes are inherent to this string. Thus, the delimiting quotes 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.

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 71.
- the **TYPE** subcommand, which establishes attributes of the file contents. For more information, see “**TYPE**” on page 87.

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 6 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 6. 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.

Notes:

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 80.

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 TCP/IP 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 TCP/IP 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
TCP/IP 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    TCP/IP    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.

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.

FTP Subcommands

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 80.

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 <i>entry type</i> followed by three bytes each of Owner, Group, and Other <i>permissions</i>
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 <i>mmm nn</i>)
time	time of last modification, provided in the form <i>hh:mm</i> (for files greater than six months old, <i>hh:mm</i> is replaced with the year in which the file was last modified, provided as <i>nnnn</i>)
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 80.
- 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 fixed-record (**F**) format minidisk and SFS files, the **size** field indicates the actual size of a file.
For variable-record (**V**) format minidisk and SFS files, the **size** field contains an *estimated* file size, this being the lesser value determined by:
 - the number of records in the file and its maximum record length
 - the size and number of blocks required to maintain the file.
 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*

Interpretation of Relative Path Information

When a z/VM FTP server processes a request to manipulate a file, for which the file or directory has not been fully qualified, the server determines where that file or directory exists by appending any supplied file or directory information to an internal *base* directory. Exactly what constitutes this base directory differs based on the file list format that is in effect for an FTP session.

When the list format is **VM**, the z/VM FTP server uses the working directory (as established by either a **CD** or **CWD** subcommand) as the *base* directory for the supplied, but not fully qualified, file or directory name. That is, the FTP server determines where the file or directory exists by appending the supplied directory or file information to the current working directory.

FTP Subcommands

For example, assume the current list format is **VM** and the current working directory is `././VMBFS:BFS:USER1/SUBDIR`. If a `GET SUBDIR2/THIS.FILE` command is supplied by a client, the FTP server will add the supplied path information to that associated with the current working directory in order to obtain a fully qualified file name. For this example, the fully qualified file for which data will be returned is `././VMBFS:BFS:USER1/SUBDIR/SUBDIR2/THIS.FILE`.

By comparison, when the **UNIX** list format is in effect, the z/VM FTP server uses only a portion of the current working directory - either the root directory, or the lowest level directory possible - as its *base* for any file or directory information supplied by a client.

That is, the FTP server determines where a file or directory exists by adding client-supplied directory or file information to the *root* directory of the current working directory. The z/VM FTP server does this in order to operate effectively with web browser clients, because these clients always provide directory and file information in a manner that is *relative* to the root directory.

Thus, when Unix-format lists are in use, the root directory for BFS is `././VMBFS:FILEPOOL:USER/`, while `FILEPOOL:USER.` is used for SFS. For minidisk and reader directories, the root directory is the minidisk or reader directory.

For example, assume the current list format is **UNIX** and the current working directory is `././VMBFS:BFS:USER1/SUBDIR`. If a client requests file data via a `GET SUBDIR2/THIS.FILE` command, the FTP server will construct and use a fully qualified *relative* path name, based on the client-supplied path information and the appropriate root directory for the file system in use (`././VMBFS:BFS:USER1` in this case). For this example, data will be returned for the file `././VMBFS:BFS:USER1/SUBDIR2/THIS.FILE`.

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”.

ACCT

```
➤—ACct—account_information—➤
```

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.

FTP Subcommands

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

APPEND

►►—Append—*localfile*—*foreignfile*—◄◄

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 41.

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 41.

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 51 and “CD or CWD” on page 54.
- 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.

ASCII

►►—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.

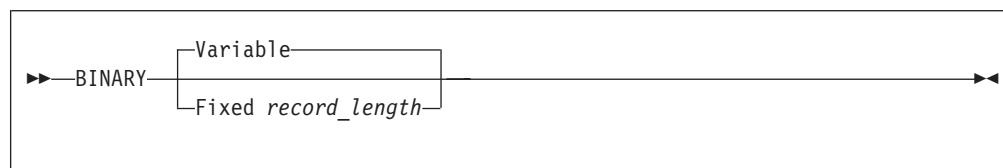
Usage Notes

- 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 6 on page 45.

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*

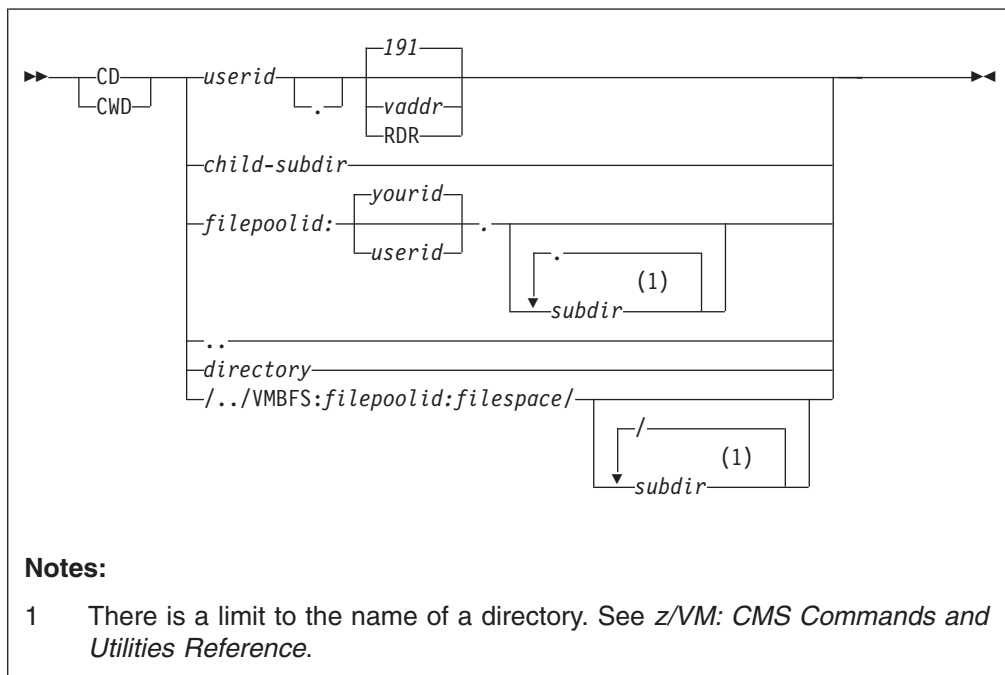
Specifies that files are stored as fixed-record format (F), with records of length *record_length*.

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 LOC SITE 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 80.

CD or CWD



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**.

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.
- When CD commands are used with a remote host running z/VM your use of SFS, BFS 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 SFS directory, the supplied directory name is treated as an SFS directory name.
 4. If a colon (:) is present in the directory name, it is treated as an SFS directory name.
 5. If the supplied directory name ends with either “ RDR” or “.RDR”, an attempt is made to change the working directory to a virtual reader.
 6. 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 subdirectory and BFS path names can present problems when it is necessary to alternate the use of SFS/BFS resources and minidisks or virtual readers as working directories. When SFS subdirectories or BFS path names exist with names that also conform to the “*userid.vaddr*” or “*userid.RDR*” directory format, that SFS or BFS resource may be obtained as a working directory instead of an intended *userid* minidisk or virtual reader. Whether this can occur is dependent upon the SFS or BFS directory hierarchy being referenced, as well as the SFS/BFS directory that is the current working directory.

For such an environment, to ensure a change to a minidisk occurs when desired, always specify a minidisk or virtual reader directory by using either the “*userid vaddr*” or “*userid RDR*” directory name format, with only spaces or blanks used as delimiters.

- The following restrictions apply to BFS files and directories:

FTP Subcommands

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 minidiskFor 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.

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.

FTP Subcommands

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.

DEBUG



Purpose

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

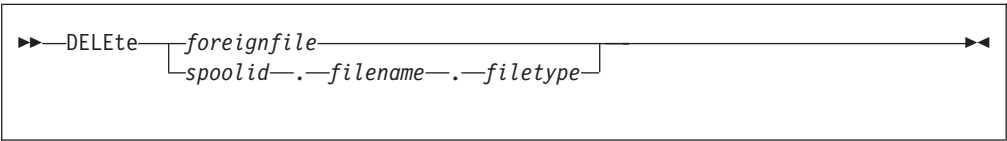
Operands

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 41.

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

►►—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

►►—Dir.

—name
—
—
—

—(—DISK—

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 41.

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 54.

- 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 or BFS directories are referenced. For other z/VM resources, responses are returned as if no operand had been specified.

FTP Subcommands

- .. 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 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” on page 23.
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 80.
- 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 67.
- 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 42.
- 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

```

>> 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 6 on page 45.

EUCKANJI

```

>> EUckanji <<
      |
      | (-NOTYPE)

```

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 335 for more information.

GET

```

>> Get foreignfile localfile (-REPLACE) <<

```

FTP Subcommands

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 41.

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 41.

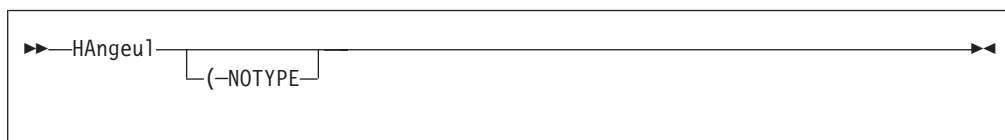
REPLACE

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 51 and “CD or CWD” on page 54 for more information.
- 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.

HANGEUL



Purpose

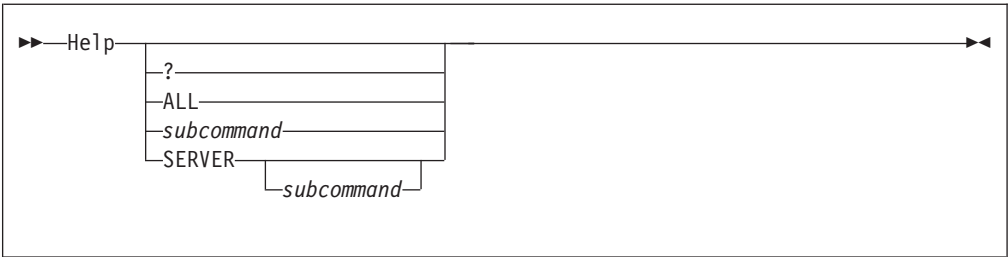
Use the HANGEUL subcommand to change the file transfer type to Hangeul.

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 335 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.

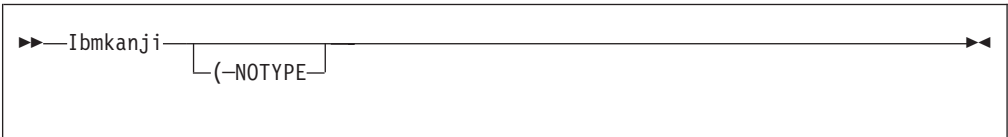
SERVER

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.

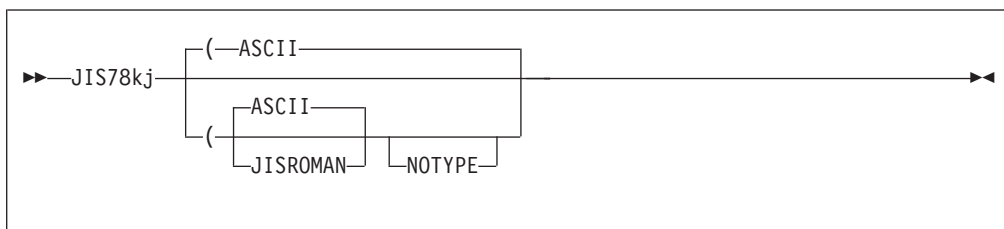
Operands

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 335 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

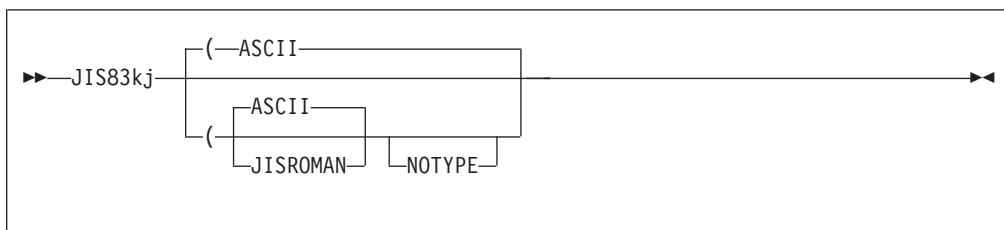
Use JISROMAN shift-in escape sequence ESC (J

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 335 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

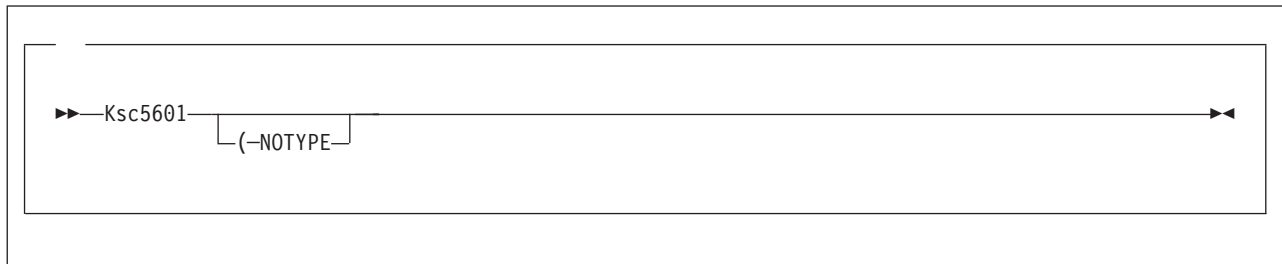
Use JISROMAN shift-in escape sequence ESC (J

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 335 for more information.

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

KSC5601



Purpose

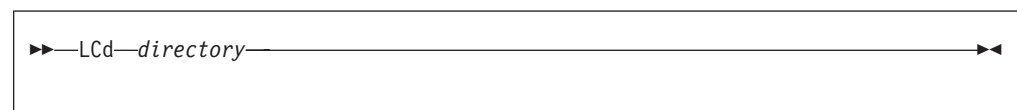
Use the KSC5601 subcommand to change the file transfer type to KSC-5601.

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 335 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.

FTP Subcommands

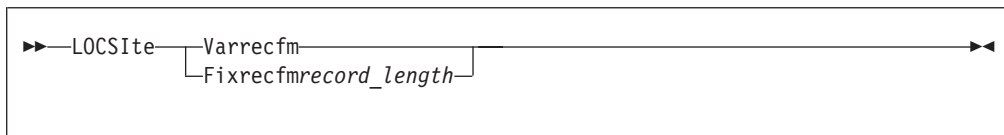
Examples

- 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.

Operands

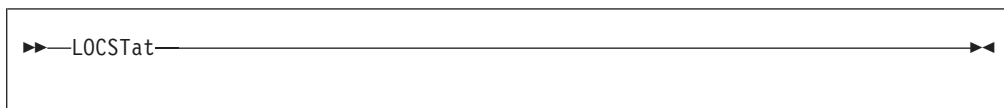
Varrecfm

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.

LOCSTAT



Purpose

Use the LOCSTAT subcommand to display local status information.

Operands

The LOCSTAT subcommand has no parameters.

The following status information is displayed:

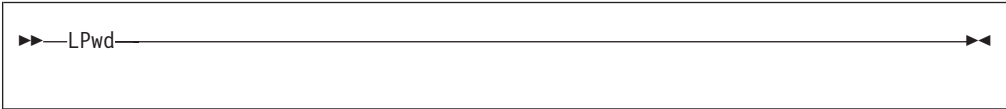
- Value of the TRACE flag (set using the FTP command or the DEBUG subcommand)
- SENDPORT setting (true or false)
- SENDSITE setting (true or false)
- Name, port number, and logon status of the foreign host
- Port number of the local host
- FTP data type (ASCII, EBCDIC, or Image) and transfer mode (block 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

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
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.

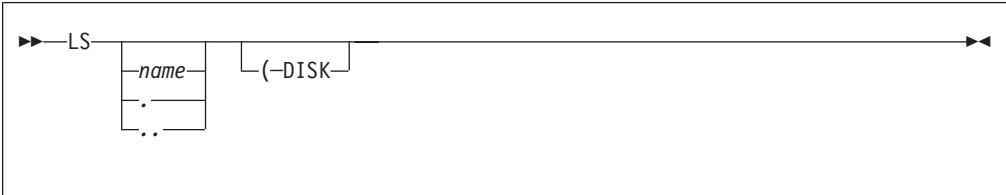
Operands

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 41.

To select which directory or file group is current, use the CD subcommand. For more information see “CD or CWD” on page 54.

- . 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 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 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 59.
- 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 42.

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
```



```

TESTFTP.B.EXEC
TRACE2.TCPIP
250 List completed successfully.
Command:

```

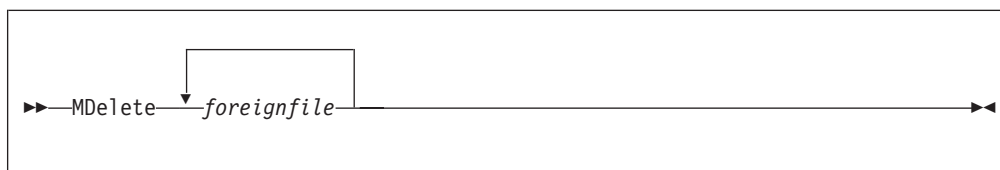
A sample LS response for a z/VM virtual reader working directory is shown here:

```

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

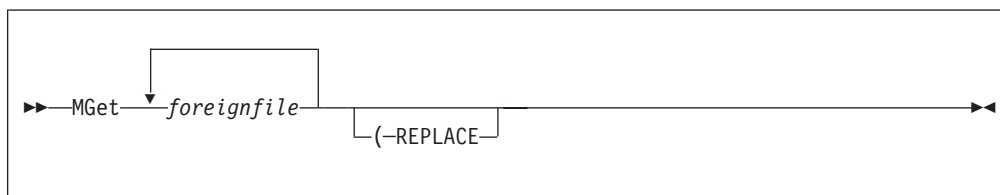
foreignfile

A foreign host file or file group to be deleted. For information about how to specify *foreignfile* see “File Name Formats” on page 41.

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 42.
- 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



FTP Subcommands

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 41.

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 51 and “CD or CWD” on page 54.
- 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 42.
- 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 62.

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

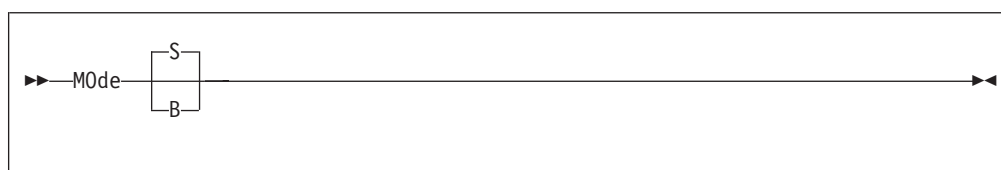
SFS permissions associated with the user ID you supply with the USER subcommand determine what directories can be created using the MKDIR command.

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 *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

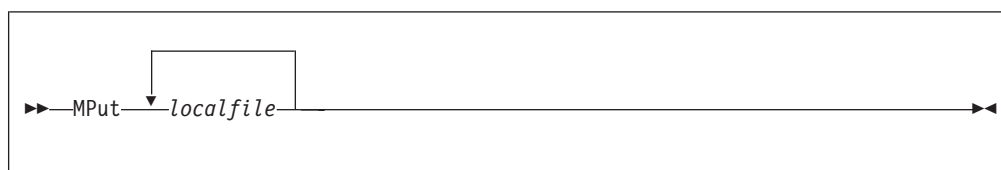
Use the MODE subcommand to define how bits of data are to be transmitted, setting the mode (data format for the file transfer).

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 6 on page 45 for more information about file transfer methods.

MPUT



FTP Subcommands

Purpose

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

Operands

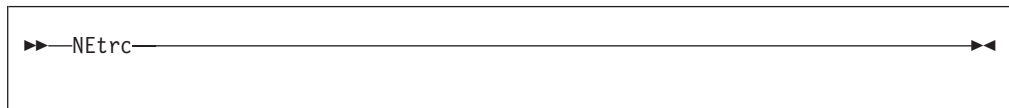
localfile

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 41.

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 51 and “CD or CWD” on page 54.
- 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 42.
- 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 75.

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.

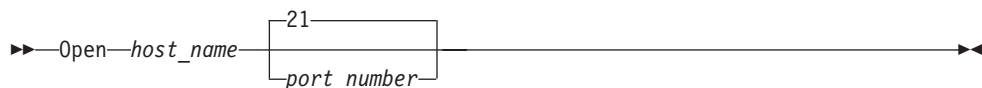
Operands

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

host_name

The host name or internet address of the foreign host.

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 57.

FTP Subcommands

- 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.
Command:
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.

PASSIVE

```
➤➤—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, data transfers are turned on and off alternately. If PASSIVE is on, the FTP client sends the server a PASV command and uses the response to determine which address and port to use when establishing the data transfer connection with the server. If PASSIVE is off, the client sends the server a PORT 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.

PUT

```
➤➤—Put—localfile—┐
                   └foreignfile┘—————➤➤
```

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 41.

foreignfile

The foreign host file to be created. For information about how to specify *foreignfile* see “File Name Formats” on page 41.

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 51 and “CD or CWD” on page 54.

FTP Subcommands

- 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 84 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.

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

```
»»—QUIT—««
```

Purpose

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

Operands

The QUIT subcommand has no parameters.

QUOTE

```
»»—QUOTE—string—««
```

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.

FTP Subcommands

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

►►—REName—*foreignfile*—*newfile*—◄◄

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 41.

newfile

The new name to be given to *foreignfile*. For more information about how to specify *newfile* see “File Name Formats” on page 41.

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 commands.

Operands

The SENDPORT subcommand has no parameters.

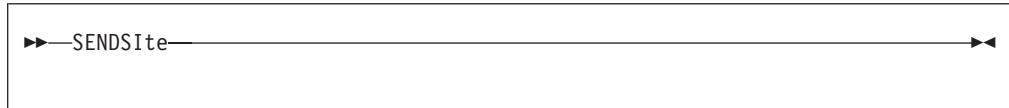
Usage Notes

- 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 command by default when establishing a connection for each data transfer. FTP does not send PORT commands for data transfer when you use the SENDPORT subcommand to disable PORT commands.
- SENDPORT is useful for communication with FTP implementations that ignore PORT commands, but show (incorrectly) that the PORT command has been accepted.

FTP Subcommands

- The SENDPORT setting is ignored when passive data transfer mode is established using the PASSIVE subcommand. This is because no PORT command is transmitted by the client in passive mode.

SENDSITE



Purpose

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

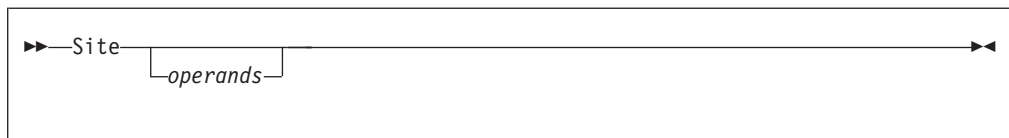
Operands

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



Notes:

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 46.

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

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. For more information about VM-format and Unix-format responses, see “File List Formats” on page 46.

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

FTP Subcommands

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. 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 *TCP/IP Planning and Customization* for further information on loading and customizing translation tables, as well as Chapter 15, "Using Translation Tables," on page 311.

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 80.
- 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

```

>> SIZE foreignfile <<

```

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 41.

The returned file transfer size may account for included record delimiter or block header bytes, depending on the current TYPE and MODE settings.

SJISKANJI

```

>> SJiskANJI [(-NOTYPE)] <<

```

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 335 for more information.

Note: JIS is the abbreviation for the Japan Institute of Standards.

STATUS

```

>> STatus [name] <<

```

Purpose

Use the STATUS subcommand to retrieve status information from the foreign host.

FTP Subcommands

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

►►—STRUCT—F—►►

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

►►—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.

Notes:

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



(-NOTYPE)

Purpose

Use the TCHINESE subcommand to change the file transfer type to Traditional Chinese.

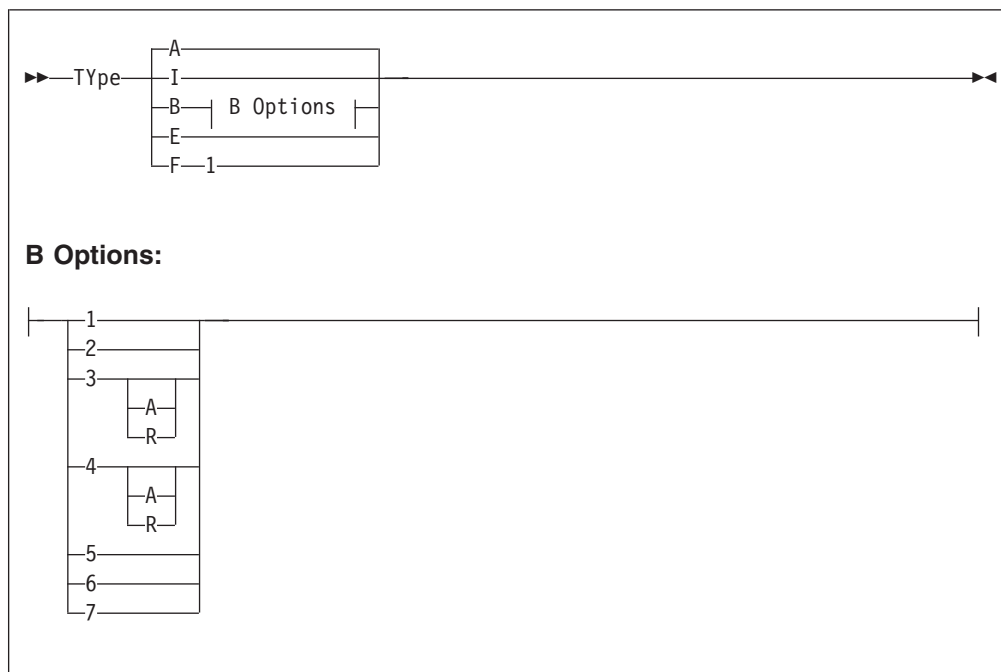
Operands

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 335 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 sever) 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 335.

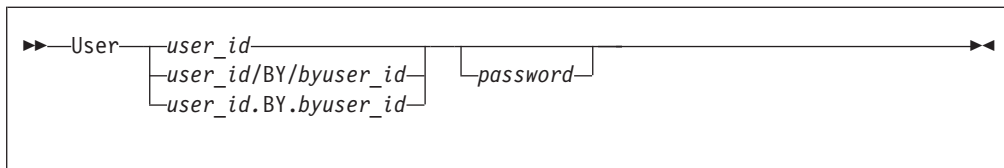
FTP Subcommands

- 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 335.

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 80.
- For more information about file transfer methods, see Table 6 on page 45.

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.

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.

Notes:

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:

Notes:

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.9 = 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.9
    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
    End

```

```

        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.

```

/* FTPFILD 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 7.

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 8 on page 94. At times XXX may instead be an FTP internal error code. For a description of possible internal error codes, see Table 9 on page 95. If an FTP server reply code or an internal error code is not available, XXX will be zero.

Table 7. 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

Table 7. FTP Command Codes (continued)

Code Number	Command	EXIT_IF_ERROR
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
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
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 8.

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 8. FTP Reply Codes

Code	Description
110	Restart marker reply
120	Service ready in <i>nnn</i> 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
230	User logged on; requested minidisk, BFS, or SFS Directory not available; proceed
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
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

Table 8. FTP Reply Codes (continued)

Code	Description
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command not implemented for that parameter
521	Directory already exists, taking no action
530	Not logged on
532	Need account for storing files
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 9 lists the possible internal error codes.

Table 9. 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	BLOCKbutNOTebcdic	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
20	AppcVMerror	false
21	SharingConflict	false

Internal Error Codes

Table 9. Internal Error Codes (continued)

Code	Error	EXIT_IF_ERROR
22	SysResrcUnavail	false
23	FSOpenError	false
24	InvalidFILEDEFDev	false
25	NoPromptConflict	true
26	InvalidArgString	true
27	NoClosingQuote	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 *TCP/IP Planning and Customization*.

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 *TCP/IP Planning and Customization*.

TFTP Subcommands

The TFTP subcommands and their parameters are described in detail on pages 98 through 102.

GET

►► Get—foreignfile—localfile—►►

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

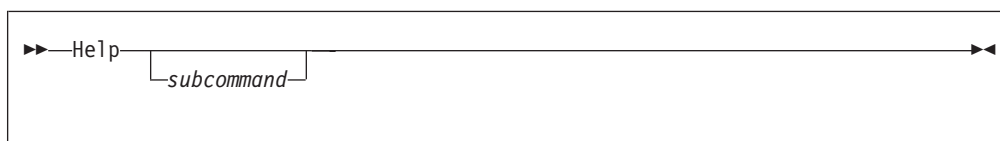
Specifies the name of the local file to be created. The *localfile* is specified by *filename.filetype.filemode* or *filename.filetype*. The default *filemode* is A.

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 100 for more information about transfer modes.

HELP



Purpose

Use the HELP subcommand to receive assistance with the TFTP subcommands.

Operands

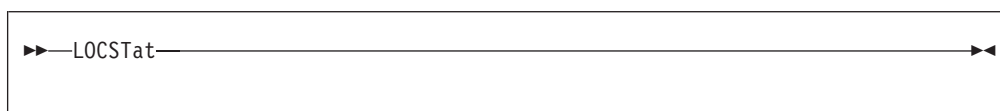
subcommand

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.

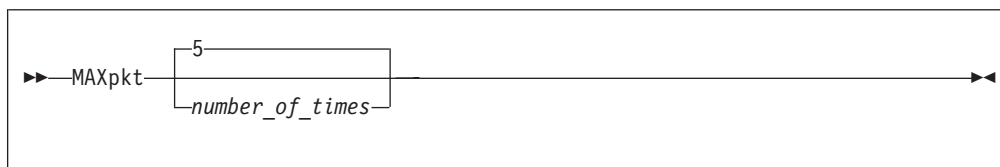
Operands

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.

TFTP Subcommands

Operands

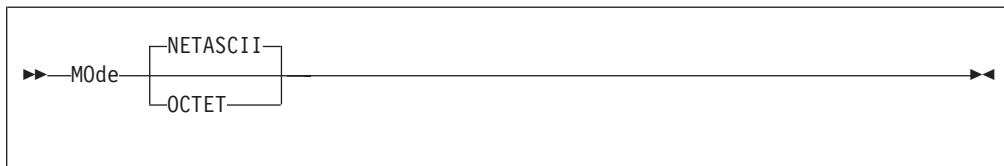
number_of_times

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 101 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

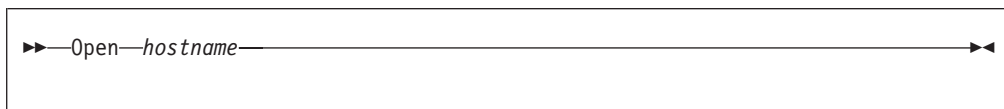
NETASCII

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 97 for more information.

PUT

►►—PUT—*localfile*—*foreignfile*—►◄

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 100 for more information.

QUIT

►►—QUIT—►◄

Purpose

Use the QUIT subcommand to end the TFTP command.

Operands

The QUIT subcommand has no parameters.

REXMIT

►►—REXmit—5
number_of_seconds—►◄

TFTP Subcommands

Purpose

Use the REXMIT subcommand to change the time-out value for each packet.

Operands

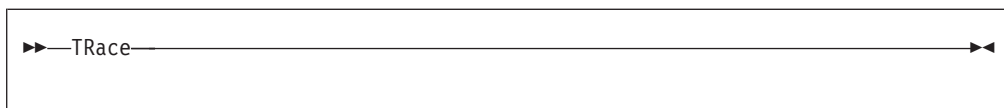
number_of_seconds

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 99 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 <i>Sending</i> or <i>Received</i> .
(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)

TFTP Subcommands

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 339.

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:

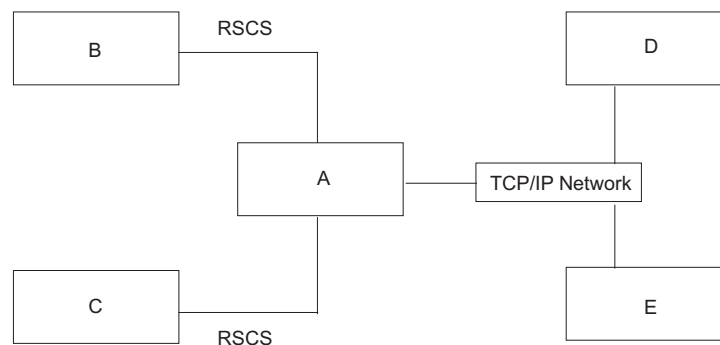


Figure 18. Example SMTP Mail Gateway Environment

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:

`user_id@A.domain`

Where:

Sending and Receiving Electronic Mail

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.

SMTPQUEUE Command



Purpose

Use the SMTPQUEUE command to have the SMTP virtual machine deliver a piece of mail that lists the mail queued for delivery at each site. The mail list is pooled to the user that issued the SMTPQUEUE command.

Operands

The SMTPQUEUE command has no parameters.

Usage Notes

1. If mail cannot be delivered to the recipients, SMTP returns a copy of the mail to the sender with an explanation of why it cannot be delivered. SMTPQUEUE queries the first SMTP server defined with the SMTPSERVERID statement in the TCPIP DATA file, or user ID SMTP on this system if none is defined in TCPIP DATA.

If mail cannot be delivered to the recipients, SMTP returns a copy of the mail to the sender with an explanation of why it cannot be delivered. SMTPQUEUE queries the first SMTP server defined with the SMTPSERVERID statement in the TCPIP DATA file, or user ID SMTP on this system if none is defined in TCPIP DATA.

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 19 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 19. 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 20 on page 108 is an example of an unknown recipient note.

Sending and Receiving Electronic Mail

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 20. 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:

<i>hostname</i>	Is the host name of the recipient.
<i>user_id</i>	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.

Sending and Receiving Electronic Mail

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 106.

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.

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 *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 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 *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

(1)

►—SMSG—*serverid*—HElp—◄◄

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 VM SMTP server.

SMSG QUEUES Command

(1)

►—SMSG—*serverid*—Ques—◄◄

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 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

(1)

►—SMSG—*serverid*—Stats—◄

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 Opns:     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
```

Sending and Receiving Electronic Mail

```
joneswk@gnylvm.vnet.ibm.com  
becomes User: joneswk  
and      Node: gnylvm
```

```
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.

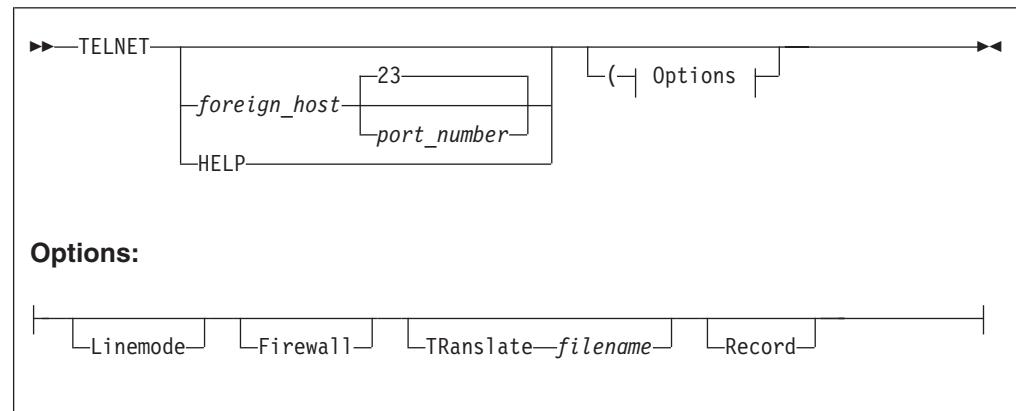
This chapter describes the following subjects:

- 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.

If you do not specify the name or internet address of the foreign host, you are prompted for the *foreign_host*.

port_number

Specifies the port number to which you want to connect on the foreign host. 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

TELNET Command

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 terminal 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.

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 119.

In Line Mode

Table 10 shows the functions that are assigned to PF keys when you invoke TELNET in line mode.

Table 10. 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 TELNET subcommands are described in detail on pages 117 through 119.

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 10 on page 116. 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

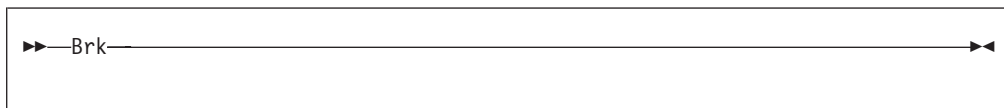
The AYT subcommand has no parameters.

TELNET Subcommands

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



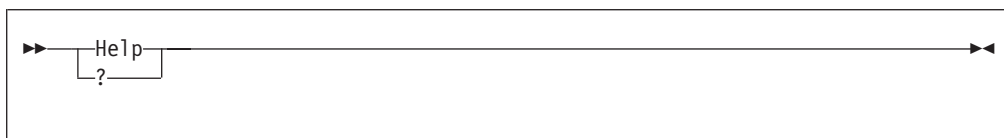
Purpose

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.

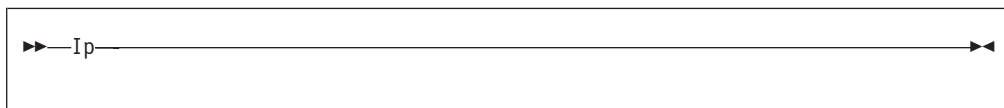
Operands

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 11.

Table 11. 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. 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 *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.

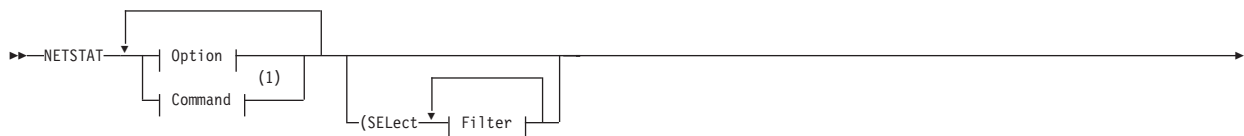
Note: For inverse name resolution, the NETSTAT command uses the ETC HOSTS file rather than the domain name server. If ETC HOSTS does not exist, the HOSTS ADDRINFO file is used.

NETSTAT Command Address Interpretation

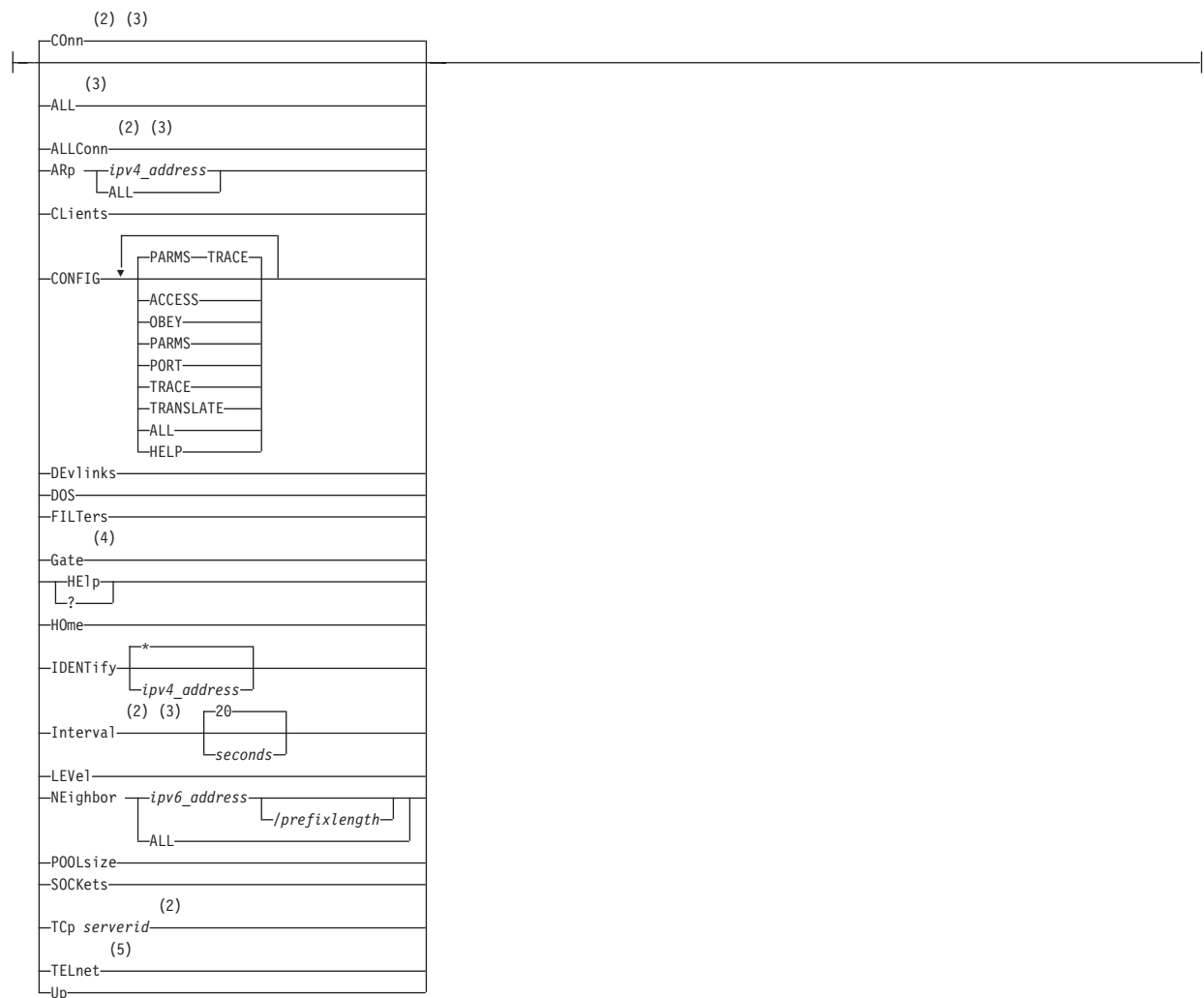
The NETSTAT command interprets and displays internet addresses symbolically, if possible. If an address cannot be interpreted, it is displayed in dotted decimal notation. Unspecified addresses are displayed as an asterisk (*).

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.

NETSTAT Command



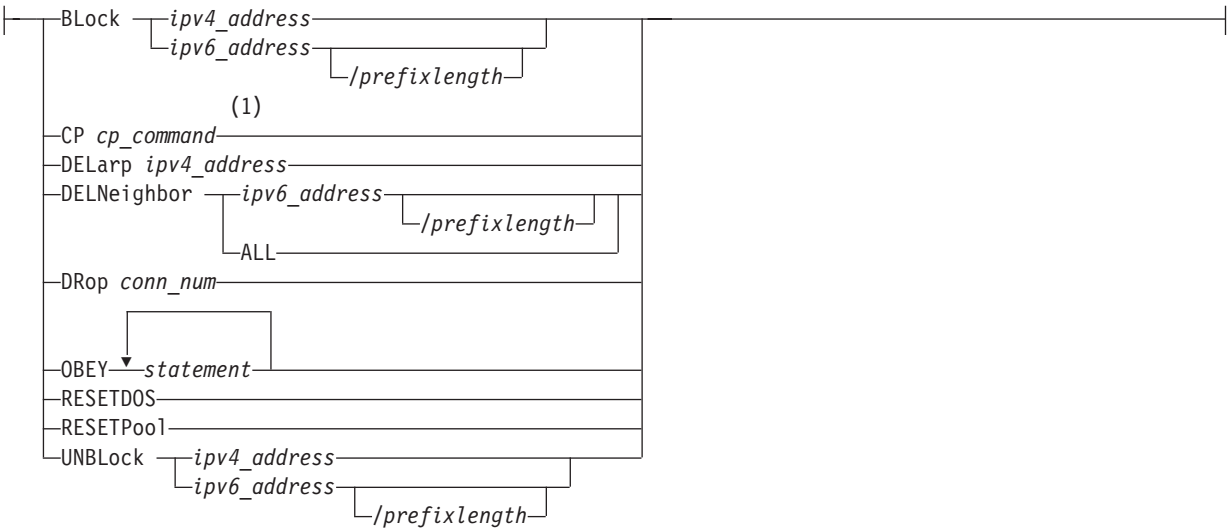
Option:



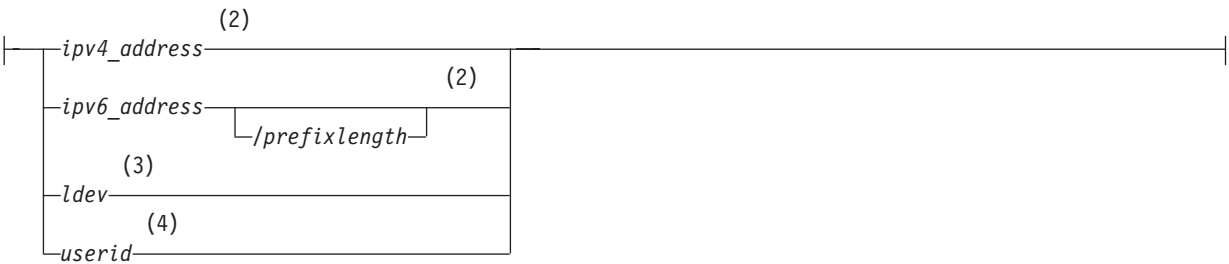
Notes:

- Only user IDs with TCP/IP OBEY command authorization can use command operands.
- Only ALLCON, CONN and TCP are valid with INTERVAL.
- The *userid* filter is valid with ALL, ALLCONN, CONN, and INTERVAL.
- The *ipv4_address* and *ipv6_address* filters are valid with GATE only.
- The *ldev* filter is valid with TELNET only.

Command:



Filter:



Notes:

- 1 All data that follows the CP keyword is used as CP command input.
- 2 The *ipv4_address* and *ipv6_address* filters are valid with GATE only.
- 3 The *ldev* filter is valid with TELNET only.
- 4 The *userid* filter is valid with ALL, ALLCONN, CONN, and INTERVAL.

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 *TCP/IP Planning and Customization*.

Displays the following information for UDP sockets being used for:

Monitoring the TCP/IP Network

- 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.

Notes:

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 which 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 which users 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.

OBEY

Displays the list of users which 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.

TRACE

Displays the current status of TCP/IP tracing. Information includes the destination of trace data, items currently being traced, as well as which users, devices, and IP addresses have been configured for selective tracing.

TRANSLATE

Displays information on internet address to MAC address translations which have been configured via the TRANSLATE statement.

Monitoring the TCP/IP Network

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 32767 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 *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.

Notes:

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:

- Device name
- Device type
- ARP query capability
- Status
- Queue size
- CPU number
- Address
- Link name
- Link type
- Net number
- Byte counts
- Packet forwarding status
- Token Ring ARP MAC address format
- Token Ring broadcast type
- VLAN ID
- Multicast specific information
- Broadcast capability
- QDIO port name and router type
- HiperSockets™ port name
- CLAW interface information
- Maximum frame size
- MTU size
- IPv6 status

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.

Some fields of the DEVLINKS display are device-dependent. These exceptions are described in the list that follows.

Monitoring the TCP/IP Network

Address	The base address is displayed for all devices.				
Byte Counts	Some device drivers do not provide counts. This is not displayed for a device type of VSWITCH-OSD.				
ARP query capability	Whether or not support for querying ARP address information is shown only for OSD or Hipersockets devices.				
Forwarding	Reports whether IP forwarding is Enabled or Disabled for the link.				
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.				
Status	<p>Some device drivers do not provide device-specific status. For these devices, possible status values are:</p> <table><tr><td>Active</td><td>The device is started.</td></tr><tr><td>Inactive</td><td>The device is not started.</td></tr></table>	Active	The device is started.	Inactive	The device is not started.
Active	The device is started.				
Inactive	The device is not started.				
Token Ring ARP MAC address format	<p>MAC address format is shown with token ring devices (either OSD or LCS). Values are:</p> <p>Canonical MAC addresses in ARP packets are converted to canonical form.</p> <p>Non-canonical MAC addresses in ARP packets are not converted.</p>				
Token Ring broadcast type	Whether broadcasts are sent to all rings (All-Rings) or just the local ring is shown with token ring devices (either OSD or LCS).				
VLAN ID	A TCP/IP for z/VM stack can be defined to be a member of a Virtual Local Area Network (VLAN). This is displayed only for QDIO Ethernet and QDIO IP links.				
Maximum frame size	The maximum frame size in use. This field is displayed for HiperSockets 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.				
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.				

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 the *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 *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. 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). |
| U | The route is up. |
| G | The route is to a gateway |
| H | The route is to a host rather than to a network. |
| C | The route was created dynamically by a redirect. |
| M | The route was modified dynamically by a redirect. |
| S | The route is a static route. Only routes defined using the GATEWAY configuration statement are flagged as static. |

Monitoring the TCP/IP Network

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.

- 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. If the Routed server is used with the BSDROUTINGPARMS configuration statement for the TCP/IP server, then these values might also differ.

- Subnet mask and subnet value (IPv4 only)

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. For more information about the HOME list (and the HOME statement), see *TCP/IP Planning and Customization*.

PI

IDENTify *ipv4_address*

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.

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.

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. Specifying **ALL** or a single asterisk (*) displays all NEIGHBOR cache entries.

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.

NEighbor ALL

Displays the neighbor cache entry for all IPv6 addresses. An asterisk (*) can be used for this purpose as well.

OBEY *statement*

Processes one or more TCP/IP server 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 *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 PROFILE TCPERROR is sent to your reader containing a description of the error and the following error message is displayed:

Configuration error. Details are in PROFILE TCPERROR.

Note: 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 *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.
- Number of elements allocated at server initialization.
- 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 *TCP/IP Planning and Customization*.

RESETDOS

Resets the data displayed for the Denial-of-Service (DOS) attacks.

Notes:

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 *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 *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*)

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.

Monitoring the TCP/IP Network

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 <i>State</i> field is blank, the <i>Conn</i> 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. <i>Conn</i> 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 <i>Bound to</i> field) are ignored; this allows TCP to retry the connection.

Tcp server

Identifies the TCP/IP Stack 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.

1nnnn 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 520

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

----
Client: ROUTED                      Last Touched: 0:00:03
Local Socket: *..520
Multicast: TimeToLive: 1 LoopBack: Yes OutgoingIPAddress: *
MulticastGroup IncomingIpAddress
-----
224.0.0.9 9.130.48.70
224.0.0.9 9.130.176.198
224.0.0.9 9.130.248.99
224.0.0.9 9.130.251.26
224.0.0.9 9.130.251.18
Ready;
```

ALLCONN

The following is a sample of the information that is displayed after entering NETSTAT ALLCONN.

Monitoring the TCP/IP Network

```
VM TCP/IP Netstat level 520
```

```
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 520
```

```
ARP Age: 5
```

```
Querying ARP cache for address 9.117.*
```

```
Link TR1 : IBMTR: 08005A8B322E IP: 9.117.32.15
```

```
Route info: 8220
```

```
Link TR1 : IBMTR: 0004AC20521C IP: 9.117.32.29
```

```
Route info: 0000
```

```
Link TR1 : IBMTR: 40000057FDBC IP: 9.117.32.249
```

```
Route info: 0592
```

```
Link ETH1 : ETHERNET: 42608C2CE222 IP: 9.117.176.4
```

```
Adapter-maintained data as of: 05/25/05 10:00:59
```

```
LINK OSDQDIO2 : QDIOETHERNET: 10005A998C6B IP: 9.117.176.1
```

```
LINK OSDQDIO2 : QDIOETHERNET: 0004AC7C82F5 IP: 9.117.176.245
```

```
LINK OSDQDIO2 : QDIOETHERNET: 0004AC7C82F5 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 520
```

```
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 520
```

```
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 520
```

```
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

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 520

Users restricted from using TCP/IP services:

BADGUY  HACKER  SPAMKING

Ready;
```

OBEY: The following is a sample of the information that is displayed after entering NETSTAT CONFIG OBEY:

Monitoring the TCP/IP Network

```
VM TCP/IP Netstat level 520

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 520

Assorted Parameters
  Check Consistency:      No      CLAW Double NOP:          No
  CP Dump:                No      VM Dump:                  No
  Equal Cost Multipath:    Yes     IPv6 Equal Cost Multipath: Yes
  Ignore Redirects:       No      IPv6 Ignore Redirects:    No
  ACB Cushion:            No      Forwarding Enabled:       Yes
  Warn on Level Mismatch: Yes     RFC1323 Support          Yes
  UDP Queue Limit:        Yes     Override Precedence:      No
  Permitted Users Only:   Yes     Proxy ARP:                No
  Restrict Low Ports:     Yes     Secure Local:             No
  Source VIPA:            Yes     Stop On CLAW Error:       No

Internal Client Settings
  Asynchronous Input:     No      CCS Terminal Name:        TNET
  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:         600
  TN3270E Enabled:         Yes     Use SC Exit for TN3270E:  No
  TN3270E Exit:            <none>   Transform:                 No
  Port(s):                 23

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 520

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).

TRACE: The following is a sample of the information that is displayed after entering NETSTAT CONFIG TRACE:

Monitoring the TCP/IP Network

```
netstat config trace
VM TCP/IP Netstat level 520

Trace Destination: CONSOLE

Tracing enabled for:
  ARP, A220 handler, Claw interrupt handler, Claw wait scan,
  *CCS CP System Service, TCP congestion control, Consistency checker,
  CTC handler, Device driver, External interrupt handler, From-1822,
  HIPPI handler, A220 trace routine, ICMP, Internal Telnet server,
  Internal Telnet timeout handler, Ioctl for routing,
  I/O interrupt handler, IP-down, IP-request, IP-up,
  DDN1822 I/O interrupt handler, 1822-Status, 1822-Timer,
  IUCV-API-greeter, IUCV handler, Monitor, Notify, Offload request,
  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, A220 common routine,
  To-CETI, CTC common routine, To-CLAW, HIPPI common routine,
  To-IUCV, To-IUCV signon, PCCA common routine, PCCA3 common routine,
  To-X25-ICA, To-1822, UDP-request, UDP-up, Offload process,
  ATM handler, ATM common routine, IGMP, Multicast, Dynamic Routing,
  OSD handler, OSD common routine, FPSM process, QDIO process, SSL,
  Denial of Service, IUCV interrupt director, Trace Table, Security,
  Neighbor Discovery

Detailed Tracing enabled for:
  <none>

Selective Tracing enabled for the following users/devices/IP addresses:
  TCPMAINT
Ready;
```

TRANSLATE: The following is a sample of the information that is displayed after entering NETSTAT CONFIG TRANSLATE:

```
netstat config translate
VM TCP/IP Netstat level 520
Network address translations specified:

IP Address      Network Address      Link
-----
9.67.22.4       HCH: FF0000009A05    TONETA
9.67.51.3       HCH: FF0000006702    TONETA
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 520

Usage: NETSTAT CONFIG options

Options:
ACCESS    - Query users with access to TCP/IP services
OBEY      - Query users with access to privileged TCP/IP operations
PARMS     - Query current AssortedParms and InternalClientParms
PORT      - Query reserved and secure port information
TRACE     - Query currently active TCP/IP tracing
TRANSLATE - Query IP address to MAC address translations
ALL       - Displays all of the above information
HELP      - 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 520

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 520

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 520

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 520

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 520

Device TOTCP2                Type: CTC                Status: Ready
Queue size: 0      CPU: 0    Address: 03F8
Link VCTC2          Type: CTC                Net number: 0
  BytesIn: 3356914    BytesOut: 3415727
  Forwarding: Enabled MTU: 9216
  Broadcast Capability: Yes
  Multicast Capability: Yes
  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
  BytesIn: 13965      BytesOut: 38904
  Forwarding: Enabled MTU: 1500
  Broadcast Capability: Yes
  Multicast Capability: Yes
  Group              Members
  -----
  224.67.113.10      3
  225.36.12.9        5
Link TR1              Type: IBMTR                Net number: 0
  BytesIn: 1810942099 BytesOut: 2027126394
  Forwarding: Enabled MTU: 2048
  Arp MAC Address: Non-Canonical
  Broadcast Type: All-Rings
  Broadcast Capability: Yes
  Multicast Capability: No

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, and TR1, an IBM Token-Ring link (indicated as IBMTR). The Net number (or, *relative adapter* number) for each device is 0; this indicates that each of these links are the first such links, of their respective type, defined for this device. The first link (ETH1) 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 second link (TR1) has LAN broadcast capabilities, but is not multicast-capable. TR1 has a non-canonical ARP MAC address and the Token Ring broadcast type All-Rings.

The status of both 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.

Monitoring the TCP/IP Network

```
VM TCP/IP Netstat level 520
```

```
Device IUCV1          Type: SNA IUCV      Status: Will retry connect
Queue Size: 0        CPU: 0    Vm Id: SNALNKB      Pgm: SNALINK        LU: SNALKA
Link IUCVL1          Type: IUCV          Net number: 1
BytesIn: 13965        BytesOut: 38904
Forwarding: Enabled   MTU: 9216
Broadcast Capability: Yes
Multicast Capability: Yes

Device IUCV2          Type: SNA IUCV      Status: Issued connect
Queue Size: 0        CPU: 0    Vm Id: SNALNKB      Pgm: SNALINK        LU: SNALKC
Link IUCVL2          Type: IUCV          Net number: 1
BytesIn: 13965        BytesOut: 38904
Forwarding: Enabled   MTU: 9216
Broadcast Capability: Yes
Multicast Capability: Yes

Ready;
```

In this example, the first device is IUCV1, whose device driver runs on virtual CPU 0. A connection attempt to the SNALNKB virtual machine has failed, as indicated by the “Will retry connect” status. This connection will be retried again in 30 seconds. The remote Logical Unit (LU) name for this device is SNALKA04. This link has LAN broadcast and multicast capabilities.

For the second device, IUCV2, whose device driver runs on virtual CPU 0, the TCP/IP server has issued an IUCV CONNECT. This connection is accepted by the SNALNKB virtual machine when the SNA sessions to LU SNALKC04 are established. This link has LAN broadcast and multicast capabilities.

For information about the status output for the SNAIUCV and SNALU62 devices, see *TCP/IP Planning and Customization*.

```

VM TCP/IP Netstat Level 510

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
BytesIn: 620680      BytesOut: 207120
Forwarding: Enabled  MTU: 8192
Maximum Frame Size : 16384
Broadcast Capability: Yes
Multicast Capability: Yes
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
BytesIn: 33932      BytesOut: 15534
Forwarding: Enabled  MTU: 8192      IPv6: Enabled
Broadcast Capability: Yes
Multicast Capability: Yes
Group                                     Members
-----
224.0.0.1                                1
FF02::1:FF00:1F                          1
FF02::1                                  1

Ready;

```

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. 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.

Monitoring the TCP/IP Network

VM TCP/IP Netstat level 520

Maximum Number of Half Open Connections: 2500

Attack	IP Address	Attacks Detected	Elapsed Time	Attack 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

Ready;

DROP

The following is a sample of the information that is displayed after entering NETSTAT DROP 1002.

VM TCP/IP Netstat level 520

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 520

Blocked IP addresses:

IP Address	Packets Discarded	Active For	Last 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 520

Known IPv4 gateways:

Subnet Address  Subnet Mask      FirstHop      Flgs PktSz Metric Link
-----
Default        <none>           9.60.28.1    UGS  1492 <none> STK40IPV6A
9.60.28.0      255.255.255.0    <direct>     US   1492 <none> STK40IPV6A

Known IPv6 gateways:

NetAddress: Default
FirstHop: 50C6:C2C1::9:60:28:1
Flags: UGS                               PktSz: 1492
Metric: <none>
Link: STK40IPV6A
NetAddress: 50C6:C2C1::/64
FirstHop: <direct>
Flags: US                               PktSz: 1492
Metric: <none>
Link: STK40IPV6A

Ready;
```

HELP

The following is a sample of the information that is displayed after entering NETSTAT HELP or NETSTAT ?.

Monitoring the TCP/IP Network

VM TCP/IP Netstat level 520

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 control 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 520

IPv4 Home address entries:

Address          Subnet Mask      Link
-----
9.130.240.135    255.255.255.0    T0240
9.130.198.12     <none>           ROSA

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:            T0240
Address:         50C0:C2C1::9:130:240:135
Flags:
DAD State:      Address Passed DAD
Origin:         HOME statement

Link:            T0240
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.

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 520
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
```

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.

Notes:

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	
User Id	Bytes Out	Bytes In	Local Port	State	Idle Time	Foreign Socket	
INTCLIEN	1977670	3640468	TELNET	Established	0:00:00	9.60.67.187..32813	
MPROUTE	4770372	112917660	520	UDP	0:00:04	*.*.*	
NAMESRV	40	16	DNS	UDP	0:00:11	*.*.*	
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	
MOYWI	1	1	46467	Established	0:01:24	50C0:C2C1::200:0:100:1F..1024	
MOYWI	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	*.*.*	
REXEC	0	0	REXEC	Listen	7:43:04	*.*.*	
PORTMAP	101336	157624	PMAP	UDP	7:43:07	*.*.*	
VMNFS	0	0	2049	Listen	7:43:10	*.*.*	
FTPSEVER	0	0	FTP-C	Listen	13:16:55	*.*.*	
PORTMAP	0	0	PMAP	Listen	13:25:34	*.*.*	
REXEC	0	0	RSH	Listen	14:11:33	*.*.*	
PERFSVM	0	0	8081	Listen	18:28:32	*.*.*	
NAMESRV	0	0	DNS	Listen	18:29:11	*.*.*	
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=Rgtright 11=Ip 12=Rfsh

LEVEL

The following is a sample of the information that is displayed after entering NETSTAT LEVEL.

Monitoring the TCP/IP Network

```
VM TCP/IP Netstat level 520
```

```
IBM 2064; z/VM Version 5 2.0, service level 0000 (64-bit), VM
TCP/IP level 520; 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 520
```

```
Querying NEIGHBOR cache for address *
```

```
Link: TONETA                      LinkType: QDIOETHERNET
Internet Address: 50C0:C2C1::9:60:59:14
Ethernet Address: 020957000025    State: Reachable
Type: ROUTER                      AdvAsDefaultRoute: NO
```

```
Link: TONETA                      LinkType: QDIOETHERNET
Internet Address: FE80::209:5700:100:25
Ethernet Address: 020957000025    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 520
```

```
OBEY command response is: OK
OBEY return code = 0
Ready;
```

POOLSIZE

The following is a sample of the information that is displayed after entering NETSTAT POOLSIZE.

```
VM TCP/IP Netstat level 520
```

```
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
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 520
```

```
Function performed
Ready;
```

SOCKET

The following is a sample of the information that is displayed after entering NETSTAT SOCKET.

VM TCP/IP Netstat level 520

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 520

Internal Telnet server status:

Conn	Status	Foreign Host	B out	B in	Logical device status	
----	-----	-----	-----	----	-----	
1118	Establsd	9.130.57.67	89606	10125	L0017 DIALED TO PVM	0503
1115	Establsd	9.82.1.118	1811	161	L00C1 ENABLED	
1067	Listen	*	0	0		
1345	Establsd	9.130.58.10	881941	1016232	L00D1 LOGON AS CIBULAMA	0009
1213	FIN-wait-2	9.185.67.151	162931	967		

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 520

Function performed
Ready;
```

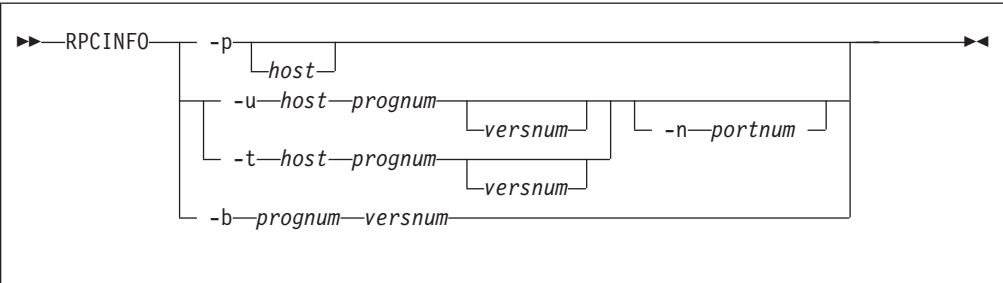
UP

The following is a sample of the information that is displayed after entering NETSTAT UP.

```
VM TCP/IP Netstat level 520

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.

Monitoring the TCP/IP Network

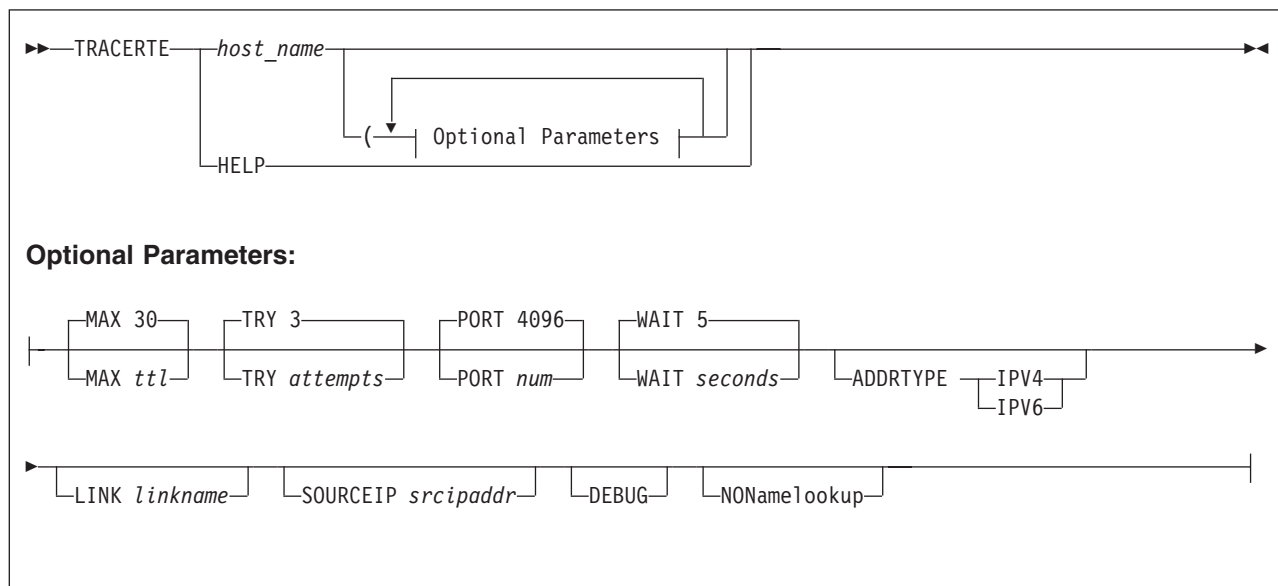
If you specify *versnum* (the version number), RPCINFO attempts to call that version of the specified program. If you do not specify a version, RPCINFO prints error information.

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

Monitoring the TCP/IP Network

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. To use the TRACERTE command (which uses rawIP functions), you must be a privileged TCP/IP user. Such users are identified with the TCP/IP OBEY statement. For more information about listing user IDs with the OBEY statement, see *TCP/IP Planning and Customization*.
4. 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.
5. 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.
6. 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*, SC24-6125.
7. 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

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

```

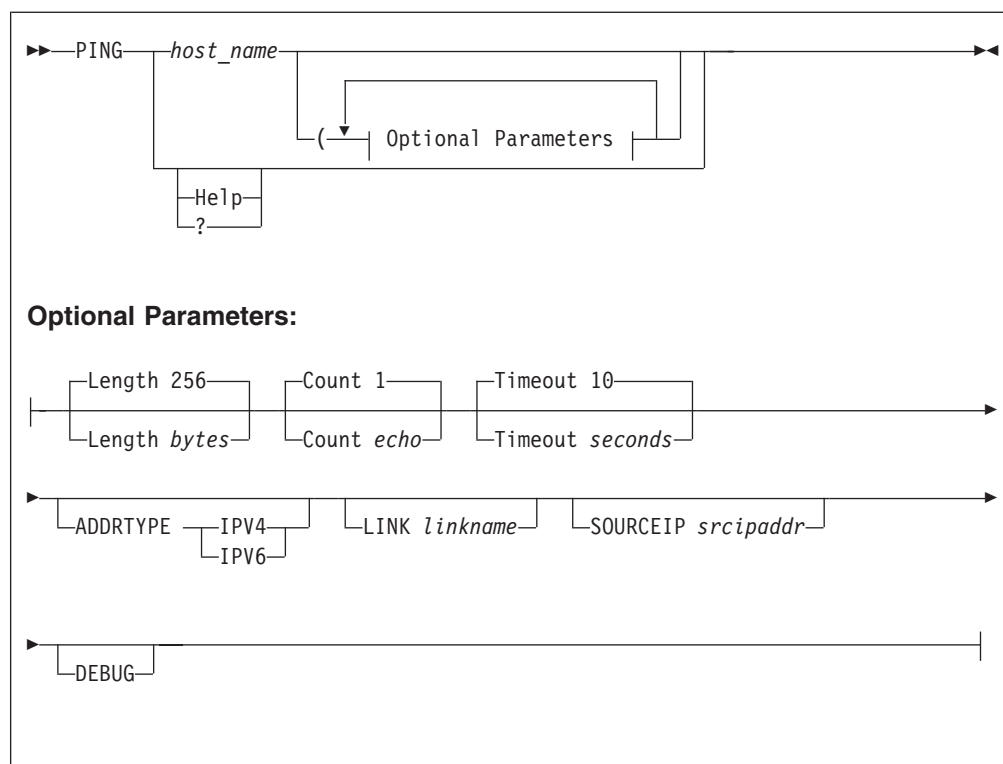
- Example with IPv6 addresses:

```

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 *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. You cannot use #CP EXT to cancel echoes from PING.

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 520: Pinging host GDLVM7 (9.56.212.11).
Enter 'HX' followed by 'BEGIN' 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 520: Pinging host LINUXIPV62.TCP.raleigh.ibm.com
(fec0:0:0:1:9:67:114:44)
Enter 'HX' followed by 'BEGIN' 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:

Monitoring the TCP/IP Network

```
ping 50C6:C2C1::9:60:28:215
Ping Level 510: Pinging host 50C6:C2C1::9:60:28:215.
                Enter 'HX' followed by 'BEGIN' to interrupt.
PING: Ping #1 response took 0.004 seconds. Successes so far 1.
Ready;
```

Chapter 7. Authenticating Network Users with Kerberos

Kerberos is a system that provides authentication service to users in a network environment. This chapter describes the Kerberos name structures and Kerberos commands. Examples of using the Kerberos commands are also provided in this chapter.

Your Kerberos system administrator can provide you with your Kerberos name and password when you register with the Kerberos system.

For more information about Kerberos, see *TCP/IP Programmer's Reference* and *TCP/IP Planning and Customization*.

Understanding Kerberos Name Structures

Before you use the Kerberos commands, you should be familiar with the structure of a Kerberos name. A Kerberos name consists of the following three parts:

Name	Description
<i>principal name</i>	Specifies the unique name of a user (client) or service.
<i>instance</i>	<p>Specifies a label that is used to distinguish among the variations of the <i>principal name</i>. An <i>instance</i> allows for the possibility that the same client or service can exist in several forms that require distinct authentication.</p> <p>For users, an <i>instance</i> can provide different identifiers for different privileges. For example, the <code>admin</code> <i>instance</i> provides special privileges to the users assigned to it.</p> <p>For services, an <i>instance</i> usually specifies the host name of the machine that provides the service.</p>
<i>realm</i>	<p>Specifies the domain name of an administrative entity. The <i>realm</i> identifies each independent Kerberos site. The <i>principal name</i> and <i>instance</i> are qualified by the <i>realm</i> to which they belong, and are unique only within that <i>realm</i>. The <i>realm</i> is commonly the domain name.</p>

Note: You must express the *realm* as a name rather than an address number. For example, use `tcp.raleigh.ibm.com` rather than `9.67.32.92`.

Kerberos Commands

To use the Kerberos commands, the KERBEROS server must be running on the same local domain or realm of one of the hosts on your network. For information about how to set up the KERBEROS server and how to use the Kerberos system administrator commands and utilities, see *TCP/IP Planning and Customization*.

The Kerberos commands for users are:

Command	Function
KINIT	Initiates a session with the Kerberos Authentication System.

Authenticating Network Users with Kerberos

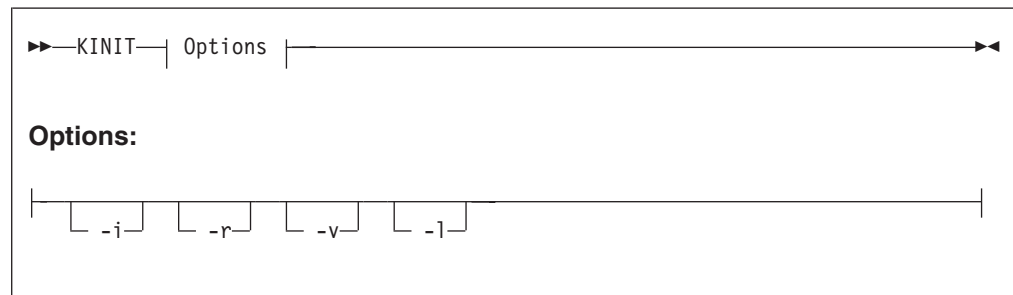
Note: The Kerberos database must be on the same realm as the client.

KLIST	Lists currently held Kerberos tickets.
KDESTROY	Destroys currently active Kerberos tickets.
KPASSWD	Modifies Kerberos passwords.

Note: The Kerberos database must be on the same realm as the client.

You enter the Kerberos commands at a VM command prompt. The following sections contain descriptions and examples of Kerberos commands.

KINIT Command



Purpose

Use the KINIT command to initiate a session with the Kerberos Authentication System.

Operands

- i Prompts you for a Kerberos *instance*.
- r Prompts you for a Kerberos *realm*.
- v Specifies verbose mode for the Kerberos response. The response contains the name of the Kerberos *realm*, and a status message indicating the success or failure of your logon attempt.
- l Specifies the TTL of the ticket, if the TTL is shorter than the TTL specified in the client, server, or database.

Usage Notes

1. You must use the KINIT command within the same realm as the database (PRINCIPL DAT/PRINCIPL IDX) used by the Kerberos Authentication Server (VMKEROB).
2. KINIT may not function correctly if the clock on the local host is not synchronized with the clock on the host running the Kerberos authentication server.
3. The KINIT command attempts to initiate a session with the Kerberos system. The KINIT command allows you to obtain an initial ticket to communicate with the ticket-granting service.

4. Use the `-i` parameter if you are registered in the Kerberos database with a non-null *instance*. For example, you may be registered as a remote administrator with an `admin instance`.
5. Use the `-r` parameter if you want to log on to the Kerberos system in a *foreign realm*.
6. Use the KINIT command without a parameter if you are an ordinary user, registered with a null *instance*, and want to log on to the Kerberos system in your *local realm*.
7. If the KINIT command is successfully processed, you are returned to a VM command prompt and an initial ticket file is saved in TMP TKT0.
8. Use the KLIST command to verify the initial ticket. The KLIST command is described in “KLIST Command.”

Using Multiple Parameters: The KINIT `-irvl` command requests the system to prompt you for an *instance* and a *realm* throughout your Kerberos session, and requests that Kerberos responses be displayed in verbose mode. Using the KINIT `-irvl` command has the same effect as using KINIT `-i -r -v -l`.

Examples

- Information displayed after invoking the KINIT command:

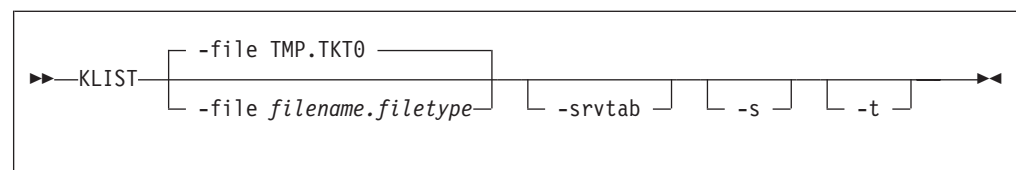
```
Kerberos Initialization
Kerberos name:
<myname>
password:
<mypassword>
```

- An example format for using multiple parameters:

```
KINIT -irvl
```

Note: When you use a Kerberos command with multiple parameters, do not enter spaces between the parameters; any entry (parameters) after a space is ignored.

KLIST Command



Purpose

Use the KLIST command to list the *principal names* of all of your current Kerberos tickets.

Operands

-file filename.filetype

Specifies the name of the file to be used as the ticket file. The default file name is TMP.TKT0, which resides on the client's 191 disk.

Authenticating Network Users with Kerberos

-srvtab

Lists the contents of the key file called ETC SRVTAB. The keys to all services provided by the same *instance* are in the key file.

-s Specifies shorthand mode for the Kerberos response. Error message and debug data will not be printed. This operand is not applicable with the -t operand.

-t Specifies TGT mode for the Kerberos response. Messages will be printed only for queries regarding Ticket-Granting Tickets (TGTs). This operand is not applicable with the -s operand.

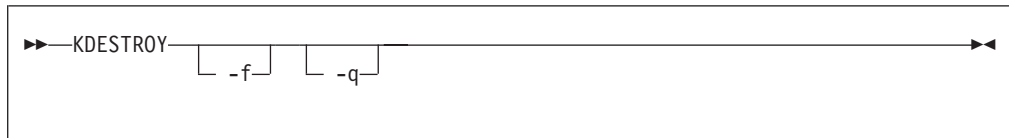
Examples

- An example of information displayed after invoking the KLIST command:

```
Ticket file:TMP TKT0
Principal:galina@univ.lab.chem
  Issued          Expires          Principal
Dec 20 13:49:16  Dec 20 21:49:16  krbtgt.univ.lab.chem@univ.lab.chem
```

If the KLIST command is not successfully processed, an error message is displayed.

KDESTROY Command



Purpose

Use the KDESTROY command to delete a currently active Kerberos tickets file.

Operands

- f** Deletes your active Kerberos authorization tickets file without displaying a status message.
- q** Eliminates the beep response to your terminal if the tickets are not deleted.

Usage Notes

1. A status message indicating the success or failure of the operation is displayed on the screen. If KDESTROY cannot delete the ticket file, the system sends a beep response to your terminal.
2. You should periodically delete your currently active Kerberos authorization tickets with the KDESTROY command. When you use the KDESTROY command, the ticket files are overwritten with zeros and removed from the system.

KPASSWD Command

```
➡—KPASSWD— -u—username— — -i—instance— ➡
```

Purpose

Use the KPASSWD command to change your Kerberos password.

Operands

- u** *username*
Specifies the full name of the user.
- i** *instance*
Specifies an *instance*, if you are registered with the Kerberos database with a non-null *instance*.
- n** *user*
Specifies the *username*.
- r** *real m*
Specify a *real m*.
- h** operand usage help.

Usage Notes

1. When you use the KPASSWD command, Kerberos prompts you for your current password. Upon verification that the current password is correct, Kerberos prompts you twice for the new password. After you enter the new password, a message is displayed indicating the success or failure of the change.

Chapter 8. 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 185.

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 186.

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 188.

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.

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. |

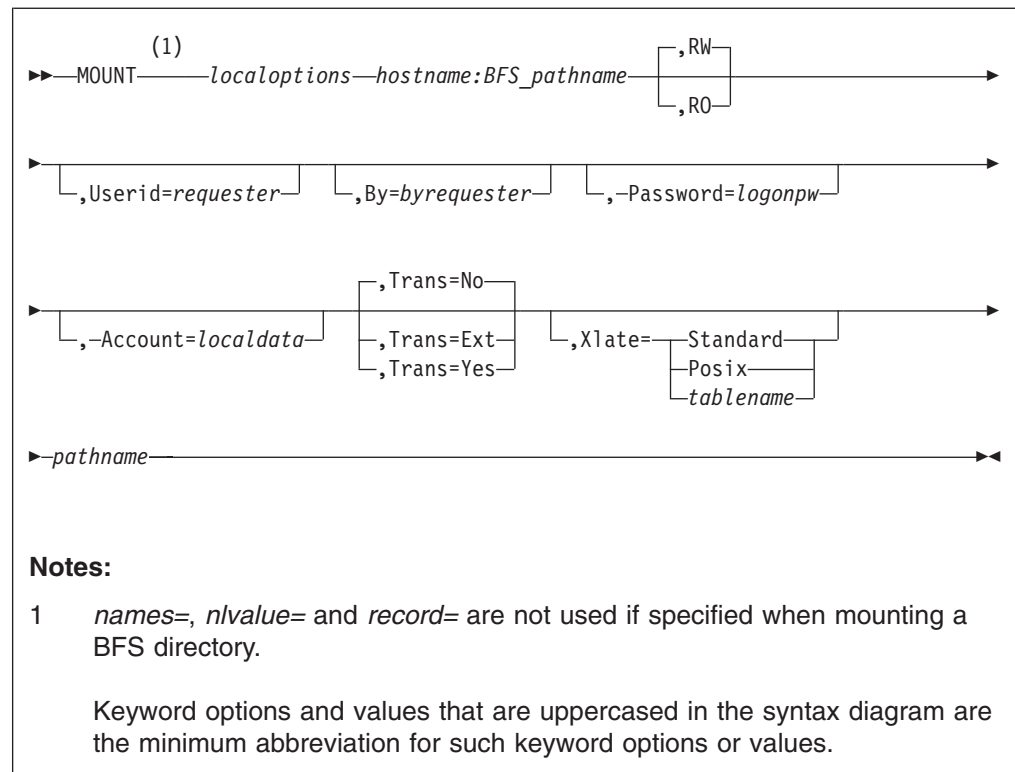
No unique error messages are associated with these commands. The error messages that result from a failed mount attempt depend on the client system, which should provide you with sufficient status information to understand why the failure occurred.

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

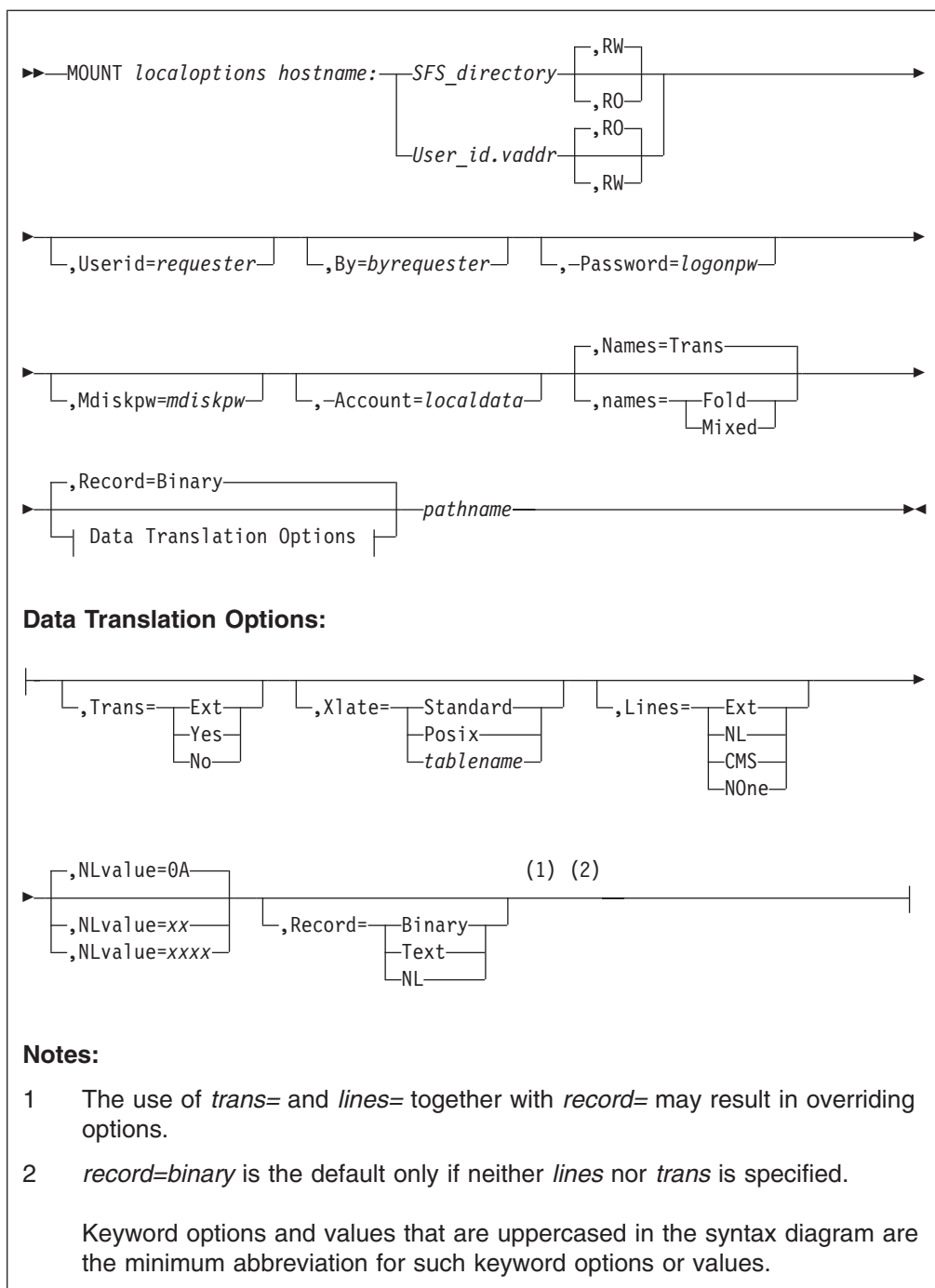
BFS file system protocols are similar to those of NFS.



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.

MOUNT Command



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.

MOUNT Command

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 *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 manager (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 12 on page 178.
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.

MOUNT Command

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').
<i>tablename</i>	The name of the translate table to be used. Some examples of <i>tablename</i> include "Xlate=UK," "Xlate=French," or "Xlate=10471252." If xlate is not specified, a system-defined translation table is used. <i>tablename</i> 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 12 on page 178.
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 21 on page 177 illustrates the MOUNT command.

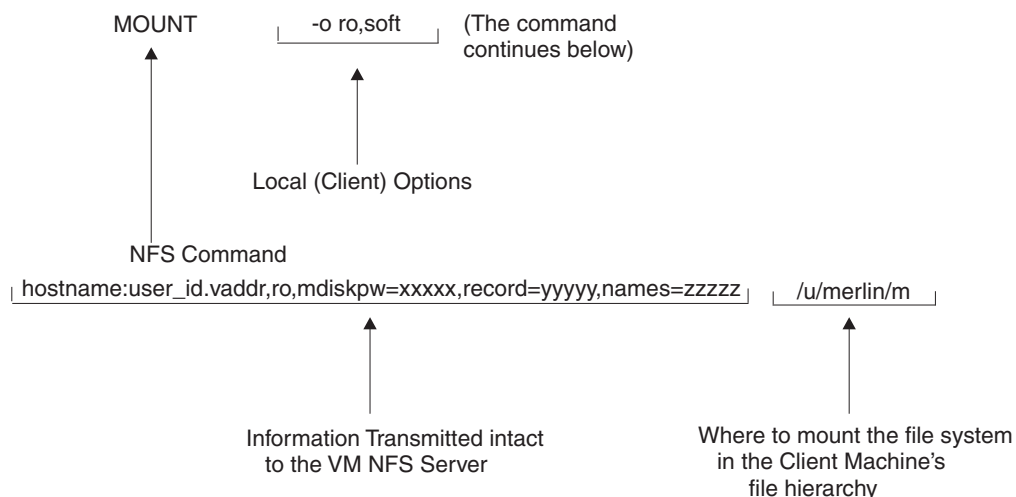


Figure 21. NFS MOUNT Command

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.

MOUNT Command

Table 12. 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		
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

Notes:

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 uppercased 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:

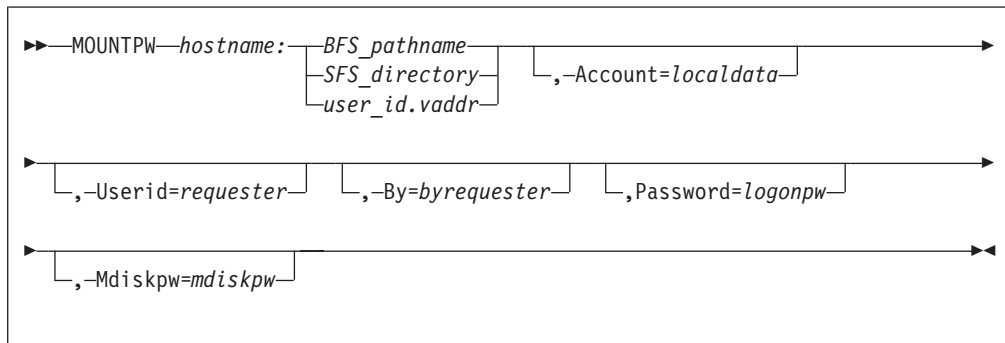
```
mount -v x: gd3vm0:elwood.191,ro,trans=yes,lines=Nl
```

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 170 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 13

Table 13. 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

```
►►—UMOUNT—pathname—————►◄
```

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 22 on page 182 illustrates the NFS UMOUNT command.

UMOUNT Command

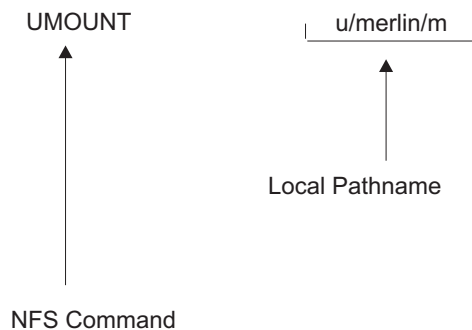


Figure 22. NFS UMount Command

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 k321sf01
```

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 k321sf01 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 k321sf01 portmapper
program 100000 version 2 ready and waiting
Ready; T=0.04/0.08 16:46:40

* See if nfs is running
rpcinfo -u k321sf01 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 k321sf01 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 14 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 14. 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.

UMOUNT Command

Table 14. 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	<p>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.</p> <ol style="list-style-type: none">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. <p>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.</p>
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 14. 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_INVALID 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_INVALID 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 supplementary 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*, SC24-6108.
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 198 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 linksThose 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.
 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*, SC24-6079, 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.

UMOUNT Command

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.

SMSG Interface to VMNFS

The following are the SMSG commands supported by the NFS server.

SMSG DETACH Command

►►—SMSG—*server_id*—*replytag*—Detach—*user_id.vaddr*————►►

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.

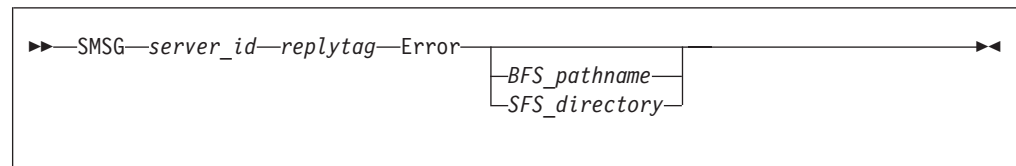
user_id

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.

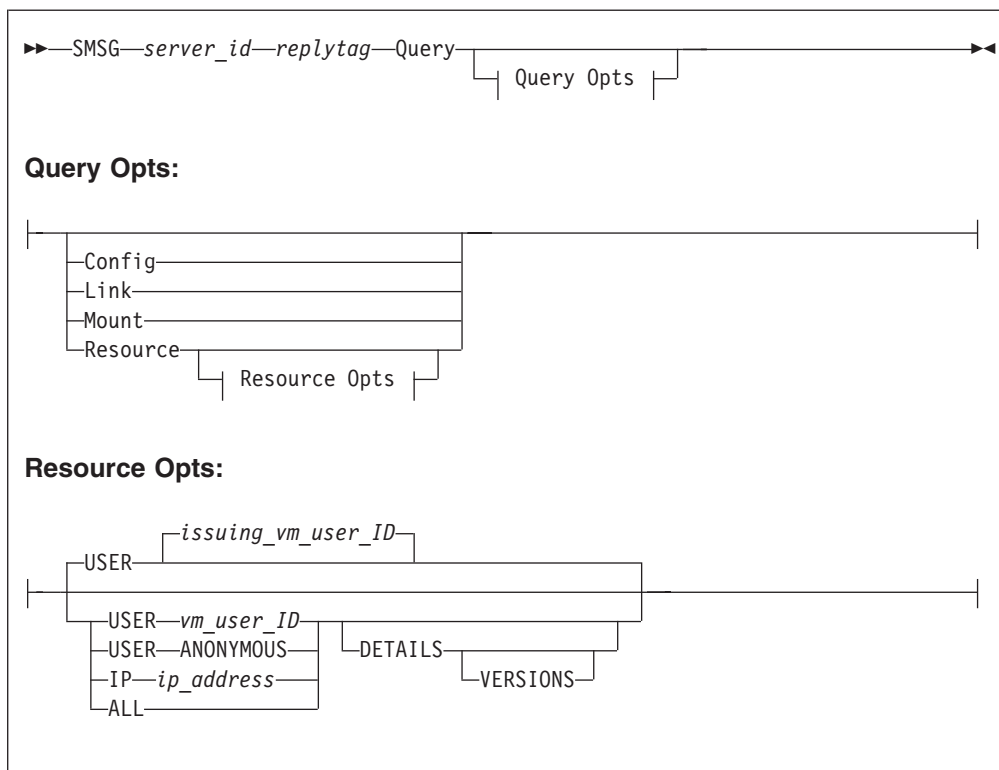
SFS_pathname

The mounted SFS directory for which host error information is requested.

SMSG Interface to VMNFS

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 precedes 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 15 shows the information that is displayed when the ALL operand is used.

Table 15. 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.
<i>name</i>	The name of the mounted BFS directory, SFS directory, or minidisk.
<i>IP address</i>	The IP address from which the mount was requested.
<i>user ID</i>	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

```
►►—SMSG—server_id—replytag—Refresh—user_id.vaddr—◄◄
```

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.

SMSG Interface to VMNFS

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: <http://www.ibm.com/s390/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 *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.

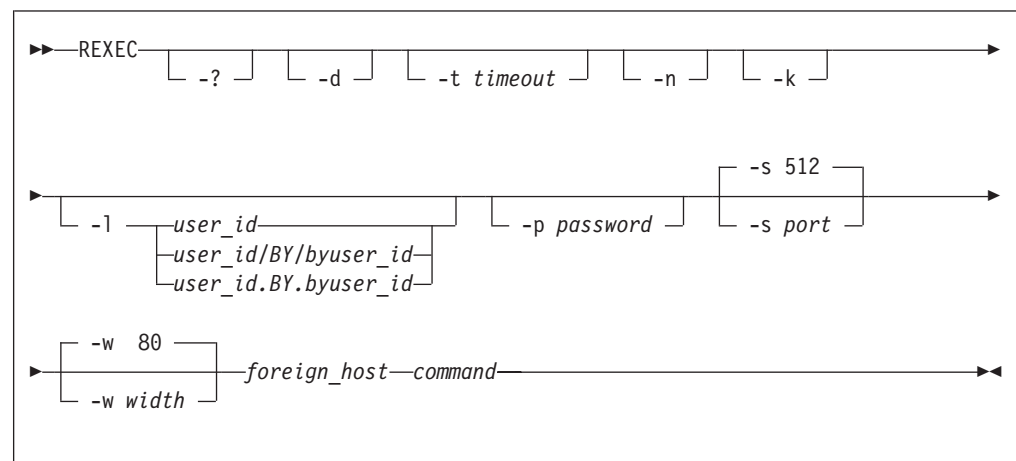
Chapter 9. Using the Remote Execution Protocol

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 *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.

-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.

Using REXEC

- 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 also be case sensitive.
 - 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
LPSEVERE - 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 327.

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.

Using REXEC

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 machine 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
```

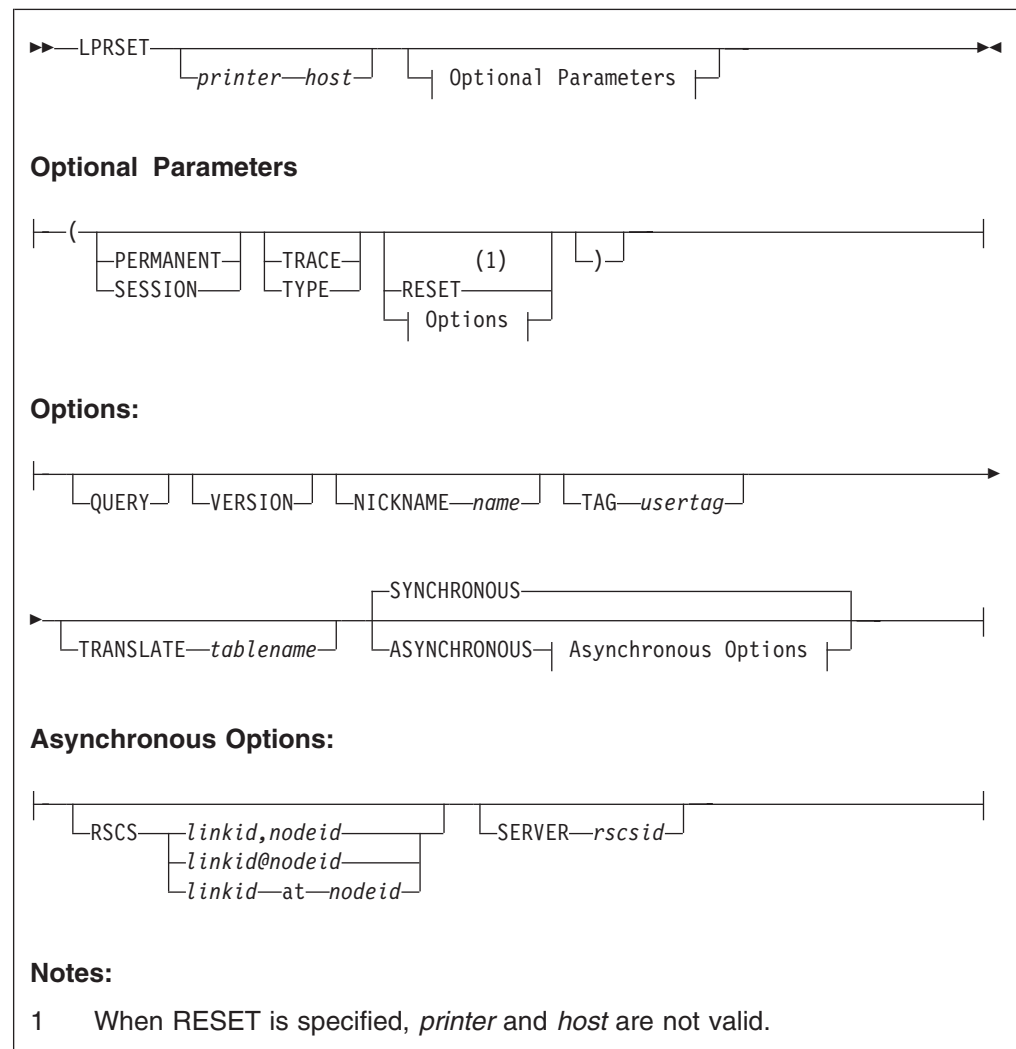

Chapter 10. Using Remote Printing

TCP/IP provides client and server 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



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.

Remote Printing Commands

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 218 and “Usage Notes” on page 218 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.

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 219.

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.

Remote Printing Commands

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 prtshr, enter the following command:

```
lprset LPTQ1 prtshr
```

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 prtshr (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

►► LPR *filename filetype* A
*
filemode

(SYNCHRONOUS Options ASYNCHRONOUS Options Syn/Asyn Options)

Syn/Asyn Options:

PRINTER *printer* HOST *host*
AT *host*
@ *host* NICKNAME *name* TAG *usertag* (1)
NOCC
CC

CLASS *mode_dflt* CLASS *data* COPIES *1* COPIES *nn* FILTER *code* LANDSCAPE TRACE
TYPE

TRANSLATE *tablename* VERSION

Synchronous Options:

ACK NOBINARY BURST HEADER JOB *filename filetype*
NOACK BINARY NOBURST NOHEADER INDENT *nn* JOB *data*

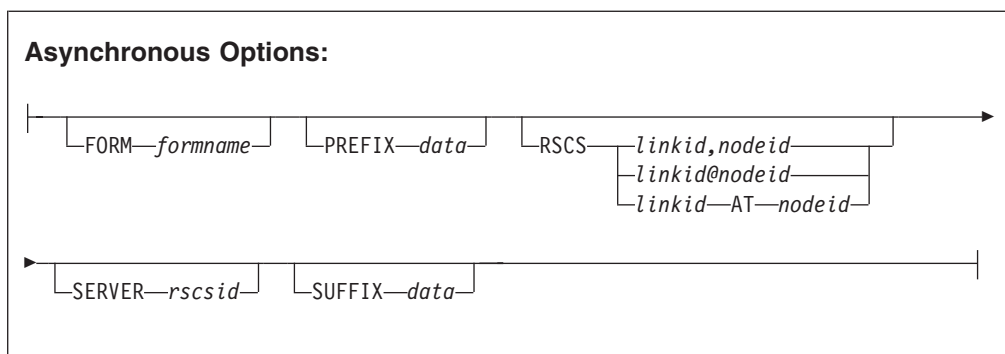
JOBNUM *nnn* JNUM *nnn* LINECOUNT *55* LINECOUNT *nn* MAIL NAME *filename filetype* NAME *data* POSTSCRIPT
NOPOSTSCRIPT

RFCports TITLE *title* WIDTH *nn*

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.

Remote Printing Commands



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 requests directed to a VM LPD server, *data* may be used as a spool file class.

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:												
	<table> <tr> <th>Character</th><th>Printing Action</th></tr> <tr> <td>Space</td><td>Start a new line of output</td></tr> <tr> <td>+</td><td>Overprint this line on the previously printed line</td></tr> <tr> <td>–</td><td>Produce two blank lines, then begin a new line of output</td></tr> <tr> <td>0</td><td>Produce one blank line, then begin a new line of output</td></tr> <tr> <td>1</td><td>Start new page, and begin printing at first line</td></tr> </table>	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
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*

Remote Printing Commands

@ *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 218 and “Usage Notes” on page 218 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.

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*, SC24-6125.

If your use of LPR requires specific translations to be performed, see Chapter 15, “Using Translation Tables,” on page 311.

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 on page 213 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 **I** 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*.

Remote Printing Commands

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 be 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 **I** 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.

Notes:

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.

Remote Printing Commands

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 TCP/IP 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 (") quotes — 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

Remote Printing Commands

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 I 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 I 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 prtshr, enter the following command:

```
lpr test listing (printer LPTQ1 host prtshr
```

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 prtshr nocc
```

- If an LPRSET command was previously issued to set the printer and host defaults to LPTQ1 and prtshr (for example, LPRSET LPTQ1 prtshr), 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 Prnting 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, 3 and 4.

- 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.ADVPR1 :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.

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
```

Remote Printing Commands

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.252150532D41646F62650D0A73746174757364696374206265676
        96E2066616C7365207365746475706C65786D6F646520656E640D0A

* 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 LPRST.

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 LOCAL and 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 are:

- For LOCAL services: FOR, PASS, DEST, and DIST
- For RSCS services: 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.

For both LOCAL and RSCS services, you can use either of the following JOB options.

Option	Description
FOR= <i>user_id</i>	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= <i>password</i>	Specifies the password for <i>user_id</i> . The default is no password. This option is required only if the RACF option has been specified for the designated service.

For LOCAL services, you can use the following additional JOB options.

Option	Description
DEST= <i>name</i>	Specifies the destination printer name or destination punch name for the job. The default is DEST=OFF.
DIST= <i>code</i>	Specifies the output distribution code. The default is DIST=OFF.

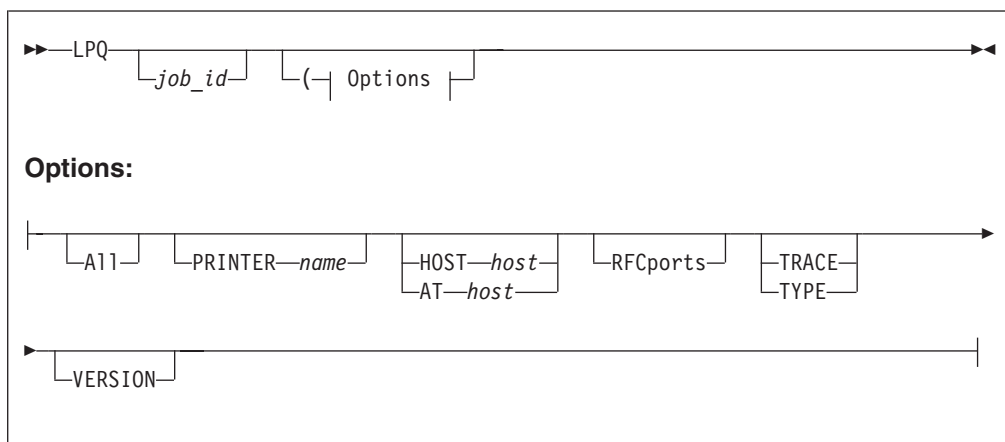
For RSCS services, you can also give any of the following: the destination node ID, user ID, and priority. These options can also be given in any order (except the OTHERS keyword, which must be last) and must be separated by commas.

Option	Description
DEST= <i>destination</i>	Specifies a RSCS destination node. The default is the node on which LPSERVE is running.
IDENTIFIER= <i>operand</i>	Specifies the second TAG operand, which can be

Remote Printing Commands

	SYSTEM, JOB, a virtual machine user ID, or a workstation node ID. The default is SYSTEM.
OTHERS= <i>option_name</i>	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.
PRIORITY= <i>nn</i>	Specifies the transmission priority. The default is 50.

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. The VM remote printing server implementation (LPD) does not support such client queries because, typically, the server does not hold print files in its internal queue. Instead, they are almost always spooled immediately to the operating system or, another virtual machine (such as RSCS) for printing. Thus, a window of time for which jobs can be queried while they are in the VM LPD print queue is almost nonexistent.
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:

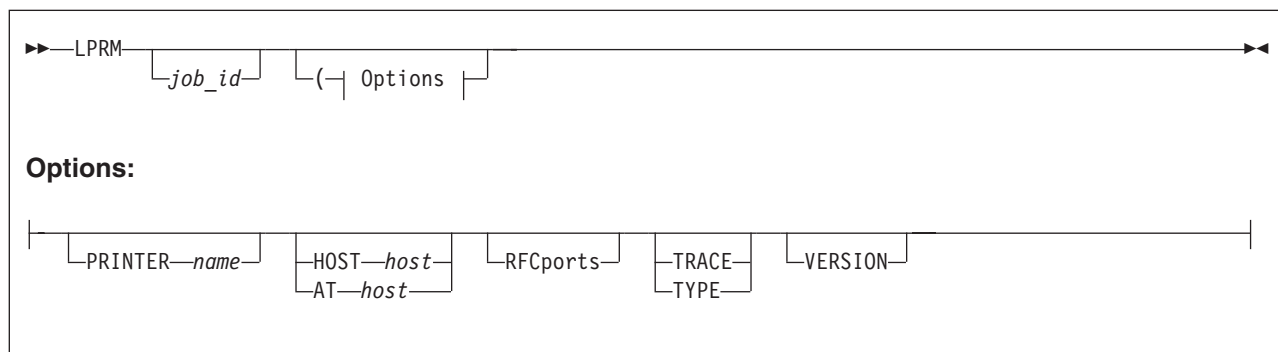
```
LPQ (PRINTER psnt HOST test1 ALL
```

This command prints a long listing of the jobs queued, including the name of the host that created the jobs.

Remote Printing Commands

- 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. The VM remote printing implementation (LPD) does not support such client requests because the server does not hold print files in its internal queue. Instead, jobs are usually spooled immediately to the operating system or to another virtual machine (such as RSCS) for printing. Thus, a window of time for which jobs can be removed while they are in the VM LPD print queue is almost nonexistent.
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 11. 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.
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

Appearance

active, the pixel spacing of an IBM 3279 Color Display Station is supplied to GDDM.

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.

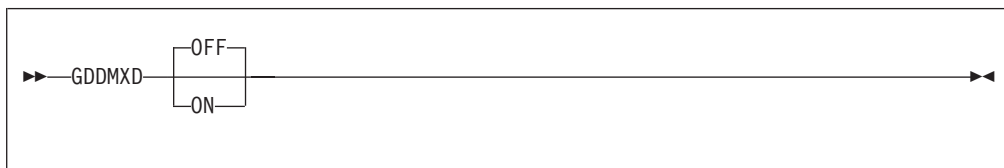
z/VM Considerations

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

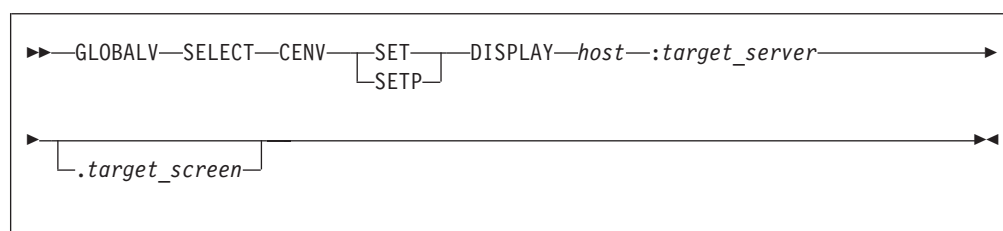
Note: If GDDM/VM Version 2 is in use, omit ADMHLIB from the above GLOBAL TXLIB command.

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.

Using GDDMXD/VM

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
- XCIconn
- Compr
- PDTrace
- ANFontn
- 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 16 on page 233.

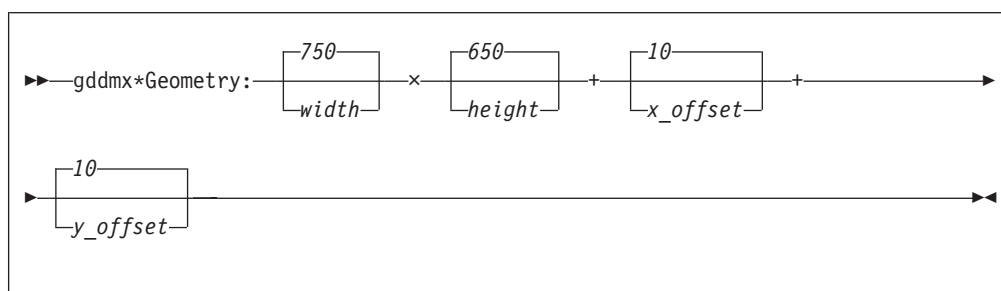
Table 16. Window Width and Window Size

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 238.

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

```
▶▶—gddmx*GColor:—nn—c————▶▶
```

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 17 lists the GDDM colors that GDDMXD/VM maps to the X Window System.

Table 17. 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
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

Table 17. *GColors* (continued)

GColornn	GDDM Color	X Window System Color
GColor14	Mustard	Wheat
GColor15	Gray	Gray
GColor16	Brown	Brown

gddmx*GMCPnn: Command

gddmx*GMCP:—nn—c

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

gddmx*ZWL:—N
gddmx*ZWL:—Y

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.

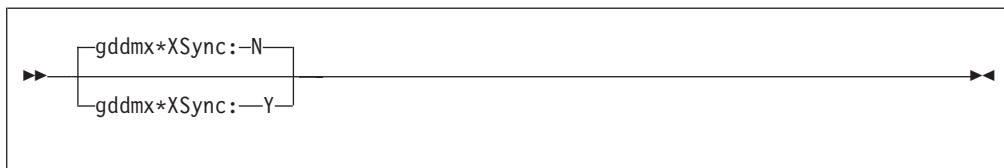
Operands

Y Directs GDDMXD/VM to use zero width lines for all drawing.

Using GDDMXD/VM

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.

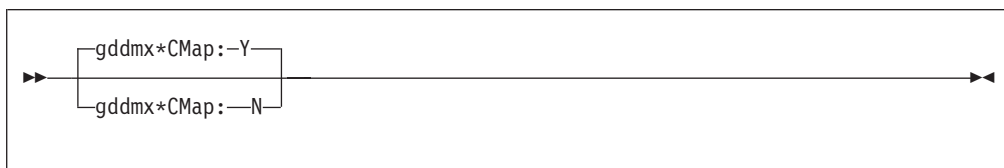
The X Window System operates asynchronously. By the time an error has been detected, the application may have issued more requests.

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.

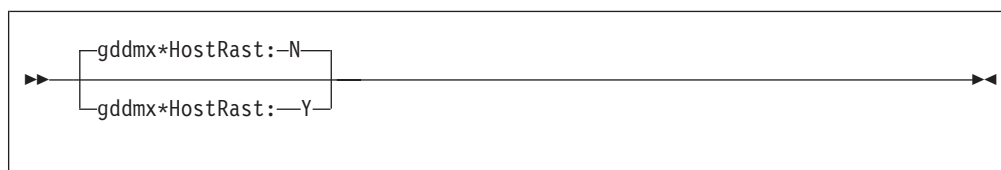
During initialization, GDDMXD/VM issues the X Window System call, `XInstallColormap`, to load the default color map (CMap). If the CMap option is specified as *N*, the `XInstallColormap` call is not made. This option accommodates the X Window System servers that load their own color maps and do not want 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.

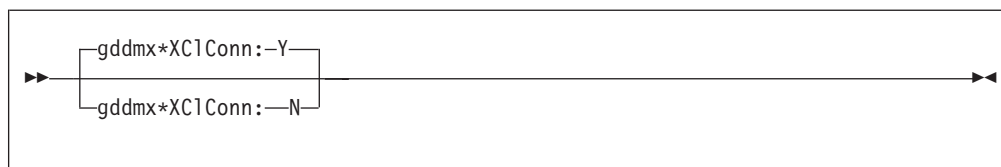
Operands

- 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*XCIConn: Command



Purpose

Use the gddmx*XCIConn command to instruct GDDMXD/VM not to close the connection to the X Window System when the GDDM device is closed.

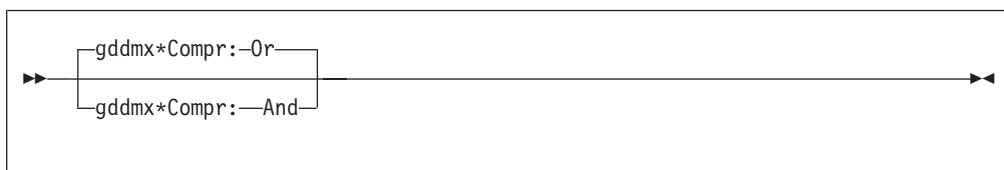
The default operation of GDDMXD/VM creates the GDDMXD/VM window when the GDDM application opens the display device, destroys the window, and breaks the connection to the X Window System display when the application closes the display device.

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 graphics window size of 480 by 352 pixels is in use.

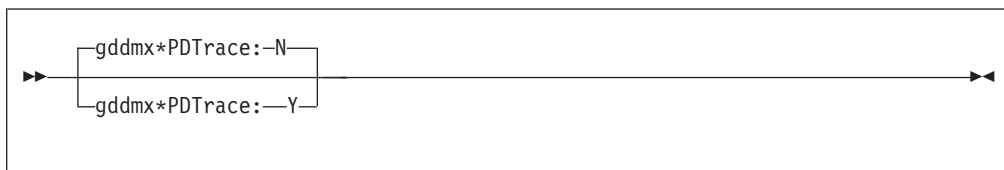
Operands

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

```
►►—gddmx*ANFont—n—fontname—◄◄
```

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 ANFontn 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 18.

Table 18. 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

```
►►—gddmx*Enter:—xxx—◄◄
```

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

Using GDDMXD/VM

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 329.

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 329.

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."

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.

```
*ADMMDF TRCESTR='FLOW'
*ADMMDF TRCESTR='PARMSF'
*ADMMDF TRCESTR='FULLTCA'
*ADMMDF 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) GDXPDT, 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 GXDEMO on

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

Using GDDMXD/VM

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 12. 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.

Notes:

1. VM SNMP supports IBM 3172 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 341.
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" 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).

SNMP/NetView Overview

Figure 23 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.

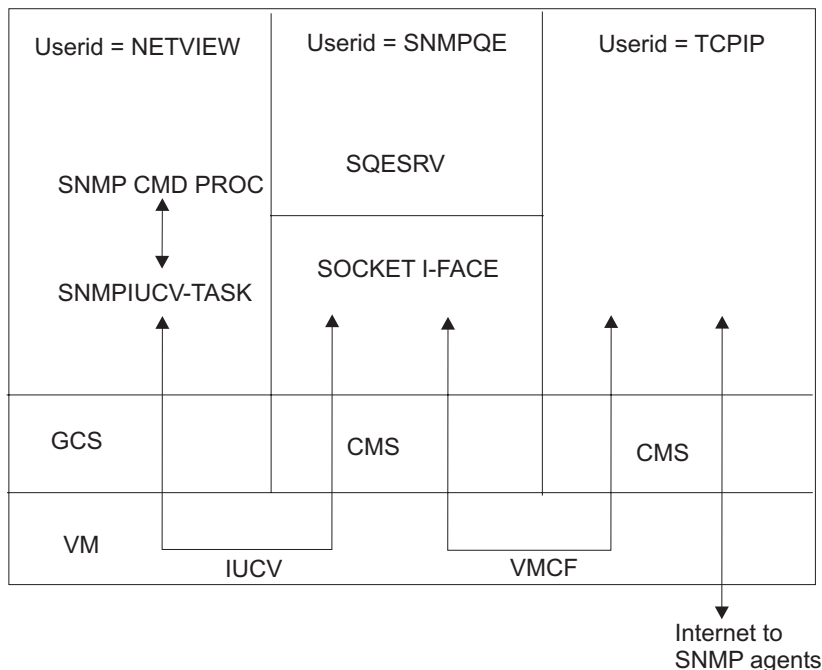


Figure 23. Overview of NetView SNMP Support

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 *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

Using SNMP

- 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

The diagram shows the command syntax for the SNMP GET command. It is enclosed in a rectangular box. Inside the box, the text is: >>—SNMP GET—host—com_name—var_name—>>. The dashes represent spaces. The host, com_name, and var_name are italicized. There are double arrowheads at both ends of the command line.

Purpose

Use the SNMP GET command to obtain one or more variables from an SNMP agent.

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, MIB-II variable names (defined in RFC 1156 and RFC 1158), and the IBM 3172 MIB variables 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 *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 261 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 259.
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 example on page 252, 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 261.
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
```

Using SNMP

```
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 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: 1.3.6.1.2.1.1.1.0
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.”

SNMP GETNEXT Command

```
►►—SNMP GETNEXT—host—com_name—var_name—►►
```

Purpose

Use the SNMP GETNEXT command in the following format to obtain the next variable in the MIB tree from an SNMP agent.

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
```

Using SNMP

- 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

```
►►—SNMP Set—host—com_name—var_name—var_value—◄◄
```

Purpose

Use the SNMP SET command to set or change the value of one or more variables in an SNMP agent.

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

```
▶▶—SNMP TRAPson—net_mask—net_desired—▶▶
```

Purpose

Use the SNMP TRAPSON command to request that the SNMP Query Engine forward SNMP traps to NetView.

Using SNMP

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 261 for information about value types, minor, and major error codes.
8. See Appendix G, "SNMP Generic TRAP Types," on page 371 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

```
►►—SNMP TRAPSOFF—filter_id—————◄◄
```

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.

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 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 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

```
➤—SNMP MIBvname—asn.1_name—➤
```

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.

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 261 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

```
►►—SNMP PING—host—◄◄
```

Purpose

Use the SNMP PING command to obtain the minimum round trip response time from the Query Engine to a specific node.

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 *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 261 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
```

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.

Using SNMP

- 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 *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.

IBM 3172 Enterprise-Specific MIB Variables

The IBM 3172 Interconnect Controller maintains a set of enterprise-specific MIB variables. The SNMP agent can act as a proxy agent to retrieve these variables from the 3172. Either a GET or GETNEXT command can be issued to retrieve the 3172 variables. The 3172 variable names can be included in a GET/GETNEXT command that also contains standard MIB variable names. See Appendix E, "Management Information Base Objects," on page 341 for a description of the 3172 enterprise specific MIB variables.

The 3172 variables are referenced by a single element instance identifier: (.1, .2, .3). For variables that pertain to the entire 3172, the instance identifier number is assigned in the order that the devices are defined in the PROFILE TCPIP file or the equivalent file in your system. For example, the first LCS device with NETMAN specified would have an instance identifier of .1, the next would be .2, and so on. No instance identifiers are assigned for non-LCS devices or LCS devices without the NETMAN keyword. For variables that are interface-specific, the instance identifier for a link is the link's ifIndex. If a GET command is issued for an interface variable using an instance identifier of a link that does not support the 3172 variables, a response of NO SUCH NAME is returned from the SNMP agent. If a GETNEXT command is issued, the links that do not support the 3172 variables are skipped over and the NEXT link that does support 3172 variables is returned.

If an error occurs accessing a 3172 variable from the 3172 (either a bad return code is received from the 3172 or no response is received from the 3172), an error code of GEN ERROR is returned to the client in the SNMP response PDU for that variable. An error message containing more specific information about the error that occurred is written to the TCPIP virtual machine (either to the console or to the trace file, depending on the specifications of the PROFILE TCPIP file). Several of the potential error conditions reference the 3172 MIB variable by the 3172 attribute index. See Appendix F, “IBM 3172 Attribute Index,” on page 369 for a list of the 3172 attribute indices and the corresponding MIB variable names.

Chapter 13. Using the Network Database System (NDB)

This chapter describes the Network Database System (NDB), which is used for relational database systems in a TCP/IP environment. NDB uses the Remote Procedure Call (RPC) protocol to allow inter-operability among different database systems. This chapter also describes how NDB is used with Data Conversion, Security, I/O Buffer Management, and Transaction Management.

The Network Database System provides :

- Interoperability through access to a mainframe relational database from AIX on RISC System/6000 or Sun Microsystem workstation.
- Ability to issue SQL statements interactively, or to imbed SQL statements within an application program.
- Client/server capabilities by providing a way for a variety of workstations users to take advantage of relational technology.

The components of the Network Database System are illustrated in Figure 24.

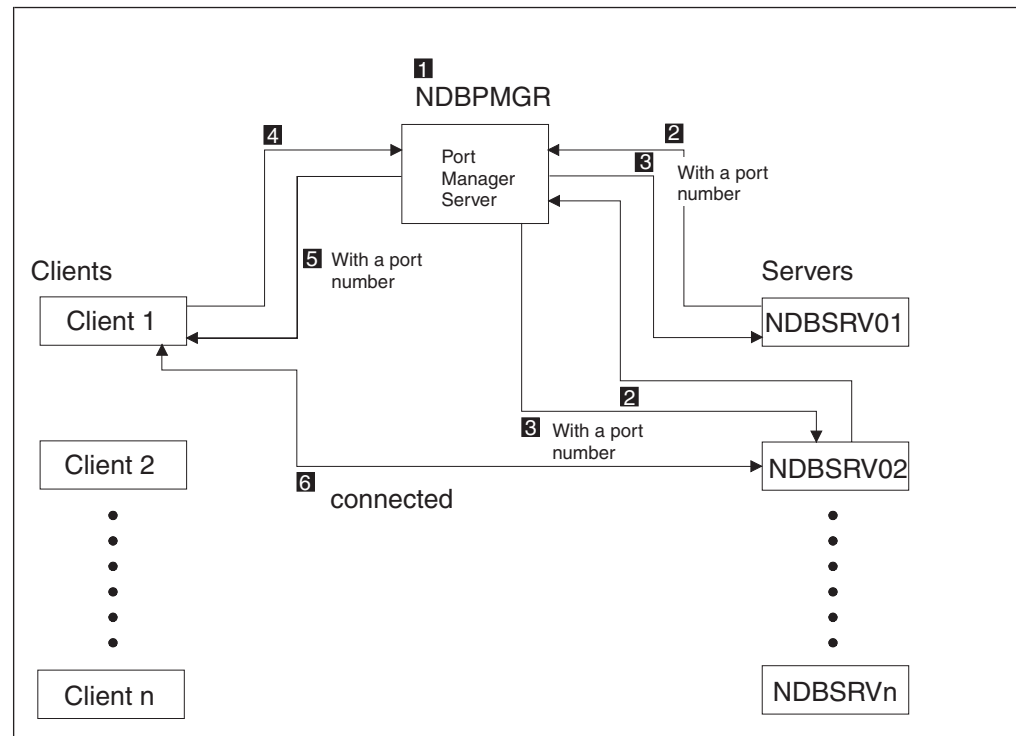


Figure 24. Components of the Network Database System

The following is a list of steps explaining the process of connecting the client to the NDB server:

1. Bring up the Port Manager NDBPMGR.
2. NDBSRV n issues a request to NDBPMGR for a port number.
3. NDBPMGR updates its port status and returns a port number.
4. A client issues a request to NDBPMGR for the port number of an available server.
5. NDBPMGR returns an available port number.

6. When the client issues a request from now on, it is connected to that NDB server, NDBSRVnn.

The Client Machine

The client machine provides mainframe or workstation applications on the Network Database system, using:

- | | |
|---------------------------------|--|
| Interactive SQL Commands | You can type in SQL commands directly from the client machine to query or access data from a database. |
| Imbedded SQL statements | You can write application programs in C and imbed SQL statements in the program. SQL statements are processed the same way interactive SQL commands are processed. |

Components of the Client Machine

The components of a client machine include end-user applications, an NDB client, and an RPC client.

- | | |
|---------------------|--|
| Applications | End-user applications run on a workstation, and can imbed SQL queries. The workstation runs the requests as its own tasks. |
| NDB client | The NDB client receives requests from the end-user application, and determines which calls to issue to the server. The NDB client packages requests into a standard format called NDBC main control block, sets up an I/O buffer, and issues RPC calls to deliver the requests to the network. |
| RPC client | The RPC client delivers calls from the client machine to the server machine. |

The NDB Client

The NDB client gets control from the application, parses the input arguments, and sets up the main control block and I/O buffers. The NDB client packages a call for the RPC client, passes control to the RPC Client, and waits for the results.

The NDBCLNT command directs the client module to deliver a request to the remote database system. The NDBCLNT command is sent to the server as a Unit Of Work (UOW). UOW is a mark for the server to recognize the threads that come from clients. A client starts a UOW by issuing a "begin" statement. The client module then requests the user to enter a valid user ID and password for a virtual machine on the remote database system. This user ID and password pair is used for authorization purposes. Both the user ID and password must be typed in; however, the password is not shown on the screen. After establishing a UOW by means of a "begin" statement and the authorization checking, the client can enter as many SQL statements as necessary. The client issues an end statement to end the UOW. Only the first request in a UOW requires the user to supply the user ID and password for a virtual machine on the remote database system. The information is stored after the first request and destroyed when the last request is issued. If a client sends only one SQL request, a UOW does not have to be started. The client simply sends one request, and the server treats the request as an entire UOW. The user ID and password of a virtual machine on the remote database system are still required in this scenario.

The RPC Client

The NDB Client uses the RPC protocol to package the request and issue a remote procedure call which sends the request to the server. The RPC client is the module that manages RPC calls and delivers calls through TCP/IP to the destination server machine. Because RPC is designed for the support of network applications, and runs with a client/server network, the RPC client allows you to avoid interfacing with the network, and provides network services without requiring any underlying network. The RPC call is transparent and independent of transport protocols.

NDB is running on top of TCP. When RPC runs on top of TCP/IP, most of the work of reliable transport is done, but the application still needs time-outs and reconnection to handle server crashes. When RPC runs on top of UDP, the application must implement its own retransmission and time-out policy.

The caller process sends a call message to the server process and waits for a reply message. The procedure's parameters are contained in the caller message. The procedure results are contained in the reply message. When the reply message is received, the results of the procedure are extracted, and the caller's execution is resumed.

RPC uses eXternal Data Representation (XDR), a data representation for machine-independent description and encoding of data.

Each RPC procedure is uniquely defined by a program number and procedure number. Several processes can be active at any time; however, each process can execute only one request at a time. The requests are executed in the order they arrive.

The RPC client sends an RPC call message to a server's PORTMAP to find a remote program's port. The server returns the relevant port number in an RPC reply message, which allows the RPC client to send the RPC message to the remote program's port.

RPC/XDR performs data conversion between different types of machines. RPC/XDR requires data type information when a call is made; however, the requester usually does not have knowledge of the data type. Therefore, RPC/XDR cannot perform data conversion between different machines. To resolve the problem, NDB makes the reply buffer a type of `string`. When data is received from the database, the NDB Server does the data conversion, and packages the result into the reply buffer.

The Server Machine

The database system must reside on the same host as the NDB server machines. The server machine has two parts: the Portmap Manager server and the NDB server. The NDB server resolves issues of security, data conversion, data buffer management, and transaction management. The Portmap Manager server provides services for clients' requests.

Components of the Portmap Manager Server

The components of the Portmap Manager server machine include:

RPC Server

The RPC server interprets an RPC call sent to the server machine from a client machine, verifies the RPC call, and distributes it to the Portmap Manager server.

Portmap Manager Server

The Portmap Manager server processes a request from either a NDB server or a NDB client. It creates or updates a status table of all program numbers used by the NDB servers. It returns an available program number to the requester. One host only needs one portmap manager server.

The RPC Server

A process is dormant awaiting the arrival of a call message. When a call message arrives, the server process extracts the procedure's parameters, computes the results, sends a reply message, and awaits the next call message. Each RPC must communicate with a dedicated port. As part of its initialization, a server program calls its host's PORTMAP to create a portmap entry for its program and version number.

The server is required to handle many client connections. The TCP server keeps the status of the open connection for each client. The number of clients is limited by the machine resources. The UDP server does not keep any status regarding the client but can handle unlimited numbers of clients.

The Portmap Manager Server

The Portmap manager server provides services for clients' requests. It has two types of clients: the NDB server and the NDB clients. The main routine, PORTMGR, maintains a table which contains the status of all the program numbers that NDB uses. The status is updated when activity occurs. It is necessary for the NDB server and the client to use the same program number so that the thread can be connected directly.

Components of the NDB Server

The components of the NDB server include:

RPC Server

The RPC server interprets an RPC call sent to the server machine from a client machine, verifies the RPC call, and distributes it to the NDB server.

Portmap Manager Client

The Portmap Manager Client issues a request to the Portmap Manager server to get an available program number, then passes the program number to the NDB server when it invokes the NDB server.

NDB Server

The NDB server is invoked by Portmap Manager Client. The NDB server preprocesses a request, sets up buffers for the result from the database, and issues a real SQL call to the database.

DataBase Utility

The Database Utility (DBU) performs SQL preprocess procedures. Each SQL statement goes through the SQL preprocess. The database processes the real request delivered from the NDB server, and returns the result to the NDB server.

The Portmap Manager Client

The Portmap Manager Client can be the NDB server or the NDB client. It sends a request to the Portmap Manager server and expects an available program number to be returned. The program number is used for future NDB requests.

The NDB Server

The NDB server provides services for client requests. The main routine, NDBSRV, receives the request from the client and determines the call type. The module performs several functions:

- Verifying the data in the control block passed from the client
- Setting up the Transaction Manager table, which contains different transactions
- Issuing an SQL call to the database
- Managing Multiple Threads, including Name Mapping, Verifying VM and SQL IDs, and Managing Unit of Work.

The NDB server is invoked by the Portmap Manager Client. If the host wants to bring up more than one NDB server, start the Portmap Manager Client on multiple (up to 20) user ID's: NDBSRV01, NDBSRV02,...NDBSR...V20.

Transaction Manager:: Most databases perform a single thread transaction, although a few databases can perform multiple thread transactions. Multiple NDB client machines can send requests simultaneously to the servers.

Unit Of Work (UOW): Unit of Work is a mark for the server to recognize the threads that came from clients. A client starts a UOW by issuing a "begin" statement. The client host machine's user ID and password are requested. The client has to type in both; however, the password is not shown on the screen. After the "begin", the client can issue as many SQL statements as it wants. The client issues an end statement to end the UOW. Only the first request in a UOW is prompted for the database machine's user ID and its password. The information is stored after the first request and destroyed when the last request is issued. If a client sends only one SQL request, a UOW does not have to be started. The client simply sends one request, and the server treats the request as an entire UOW. The host machine's user ID and password are required.

The Database Utility

NDBFRONT is the main routine. It is a front end to the SQL preprocess program and performs the following functions.

- Parsing the request that is an SQL statement
- Packaging I/O buffers
- Performing data conversions and managing different SQL data types.

The Database Server: The Database Servers that NDB utilizes are relational database management systems that manage, store, and retrieve data. This data is stored in a database that is controlled by the database server. DATABASE 2[®] (DB2[®]) is the IBM relational database management system for the MVS environment. For more information, see the *DB2 Universal Database for OS/390 SQL Reference* (SC26-9014-01).. SQL/DS[™] is the relational database management system for the VM environment. For more information, see the *IBM SQL/Data System General Information for IBM VM Systems* manual (GH09-8074)..

Connecting to a Database

There are two ways to connect to a database system:

Explicit connection	Explicit connection uses the preprocess calls, such as CONNECT, OPEN, CLOSE, DISCONNECT.
Implicit connection	Implicit connection does not use these functions, but starts making SQL calls.

Using NDB

If an SQL call occurs before the connection to the database has been established, the connection to the database is implicit, otherwise the connection is explicit. The explicit connection services are used to maximize flexibility and control.

The CONNECT function allows application programs to connect to, and use a database. It provides a call interface to the database connection services. Application programs can control the exact state of their connection to the database. It is restricted to a single task level. The application program and the database system understand the connection status of multiple tasks in an address space.

The explicit connection preprocess calls are:

Function	Description
CONNECT	Establishes a connection between the current address space and a database, and establishes the current task as a user of database services.
OPEN	Establishes the current task as a user of database services, and allocates database resources for SQL access (creates a thread).
CLOSE	Deallocates database resources (terminates the thread), and removes the task as a user of database services if the connection was established by OPEN instead of CONNECT.
DISCONNECT	Removes the task as a user of a database service. DISCONNECT also terminates the address space connection to a database if this is the only remaining task in the address space established as a user of database services.
SQL Calls	Pass through the Language Interface by branching to the entry point.

NDBCLNT Command

```
►►—NDBCLNT—machine_id—"SQL_statement"—►►
```

Purpose

Use the NDBCLNT command to direct the client module to deliver requests to the remote database system.

Operands

machine_id

Specifies the VM machine name where the NDB server is running.

"SQL_statement"

Specifies the SQL query statement.

Examples

- A single SQL query statement is issued from a client machine.

```
ndbclnt machine_id "select * from table2"
```

The client machine prompts you for a VM user ID.

Please type in your Host user id ==>

After you type in your user ID, the client machine prompts you for a password. The password you type is not displayed on the screen.

Please type in your Host password ==>

The user then waits for the result of the command issued.

If the return code is -8, the user ID and password do not match. You must enter an end command to end this UOW before you can start a new UOW.

If the return code is 25, it means that the user has accessed a table which is greater than 4096 characters. There is more data to be returned. If the user wants to have all data returned, using multiple statements in a UOW is recommended.

Multiple SQL query statements in a UOW require a "begin" and an end. The subsequent SQL query statements and the end do not require quotes.

- Multiple SQL query statements are issued from a client machine.

```
ndbclnt machine_id "begin"
```

The client machine prompts you for a VM user ID.

Please type in your Host user id ==>

After you type in your user id, the client machine prompts you for a password. The password you type is not displayed on the screen.

Please type in your Host password ==>

The client machine prompts you for the next command.

Input your SQL statement without any quotes

You enter your next command.

```
select * from table2
```

When the result of the command is returned, the client machine prompts you for your next command.

Input your SQL statement without any quotes

You enter your next command.

```
select column1 from table2
```

When the result of the command is returned, the client machine prompts you for your next command.

```
end
```

The client machine returns a message.

You are done with your Unit of Work

If the table is greater than 4096 characters you will get a return code of 25, which means there is more data. You can enter a continue command to continue to receive the rest of the table, or you can enter another SQL select statement if you do not need the rest of the table.

Format of Output Displayed on the Client

The output can be treated as records. Each record contains three fields shown below:

Field	Description
-------	-------------

datatype	char[2]
datalength	char[5]
data	char[x] where x is the content of datalength

For this release the following data types have been implemented.

Data Type	Description
li	Specifies a long integer with indicator
ln	Specifies a long integer without indicator
si	Specifies a short integer with indicator
sn	Specifies a short integer without indicator
ci	Specifies a char with indicator
cn	Specifies a char without indicator
vi	Specifies a varchar with indicator
vn	Specifies a varchar without indicator
fi	Specifies a float with indicator
fn	Specifies a float without indicator

For example, a record of the output might be as follows:

Record	Meaning
ln 3 925	Specifies a long integer without indicator. The length of the data is 3. The value is 925.
ci 10 ABCDEFGHIJ	Specifies a char with indicator. The length of the data is 10. The data is ABCDEFGHIJ.

In SQL/DS, with indicator means it allows NULL characters and without indicator means it does not allow NULL characters.

The NDB client displays only the first 360 characters. If the user wants to see the entire output, look at the output file. The default output is `ndb.output`. All the results will be appended to the output file.

SQL Commands from a Remote Machine

You can issue an SQL command from a remote machine. The syntax is:

```
ndbcInt machine_id "select tname, creator from system.syscatalog"
```

You can issue a sequence of SQL commands between "begin" and end.

In the following example, `ndbcInt` is the function name, `machine_id` is the remote machine where the database is located, and the SQL statement is between the double quotes. However, after a UOW has started, SQL commands do not require quotes.

```
ndbcInt machine_id "begin"
select * from table1
select * uid.table where col = something
:
end
```

SQL Commands from an Application

When the SQL command is called from an application, the syntax is:

```
ndbcInt ("machine_id", "select * from uid.table")
```

The explanation of each field is the same.

You can write a C language program that uses NDBCLNT as C function call. To use NDBCLNT as a C function call, enter the command in the following format:

```
ndbcInt(" machine_id ", " sql_statement ");
```

where *machine_id* is the machine where SQL/DS is running and *sql_statement* is any SQL query statement.

When you write a C language program using NDBCLNT as a C function call, you must also generate an executable module.

There are two ways to generate an executable module. The following list describes one method for generating an executable module:

- Compile your C program and NDB client source code.

If you have NDB client source code and do not have the object files, you must perform the following commands:

```
cc -c your_program.c
cc -c ndbcInt.c
cc -c ndbcancr.c
cc -c ndb_cInt.c
cc -c ndb_xdr.c
```

The *your_program* variable represents the name of your C language program containing the `ndbcInt` call.

If you have all the NDB client object files, you must perform the following command:

```
cc -c your_program.c
```

- Once you have the object files, you can generate an executable module. To generate an executable module perform the following command:

```
cc -o appl application.o
ndbcInt.o ndbcancr.o ndb_cInt.o ndb_xdr.o
```

where *appl* is your final executable module and *application* is your C language program.

- You can run your application by issuing *appl*, your final executable module.

The following list describes the second method for generating an executable module:

- Compile your C language program and NDB client source code.

If you have NDB client source code and do not have the object files, you must perform the following commands:

```
cc -c your_program.c
cc -c ndbcInt.c
cc -c ndbcancr.c
cc -c ndb_cInt.c
cc -c ndb_xdr.c
```

The *your_program* variable represents the name of your C language program containing the `ndbcInt` call.

If you have all the NDB client object files, you must perform the following command:

```
cc -c your_program.c
```

- Use the following archive command to generate a library:

```
ar rv libndbc.a ndbcInt.o ndbcancr.o ndb_cInt.o ndb_xdr.o
```

Using NDB

where *rv* are the options and *libndbc.a* is the library that is generated by the *ar* command.

- Issue a *ranlib* command to adjust the table of contents of libraries.
- Issue the following *cc -o* command to generate an executable module:

```
cc -o appl application.o libndbc.a
```

where *appl* is the executable module name and *application.o* is the object file of your C language program.

- You can run your application by issuing *appl* the executable module.

Figure 25 is an example of a C language program that has an *ndbcInt* call.

```
main(argc, argv)
int argc;
char *argv[];
{
    .
    .
    .

    ndbcInt("eduvvm2", "select * from willow.table1");
    .
    .
    .

    exit(0);
}
```

Figure 25. An example of a C Language Program

Security

There are two levels of security.

- System security

System security verifies that the clients are authorized to access the host system on which the relational database system resides.

- Database security.

Database security verifies that users are authorized to access the database. The database system administrator authorizes the different attributes to the different user IDs. For example, some users have read-write access, while others may have read-only access.

Limitations

You should be aware of the following limitations for the Network Database System on VM.

- NDB accepts query statements such as *select*. However, statements that change the content of tables are not allowed. This protects the user's data integrity.
- NDB supports only the following data types:
 - INT
 - SMALLINT
 - CHAR
 - VARCHAR
 - FLOAT
- Only C language is supported by NDB for client application program interface.

Additional information on the NDB Client code

Sample client code for a SUN workstation and for an RS/6000[®] running AIX V3.1 are provided on the TCP/IP VM product tape. For the SUN version, the three files on the tape are named:

- SDBCANCR C
- SDB_CLNT C
- SDB_XDR C

If you are using a SUN UNIX workstation as the client, rename these 3 files back to NDBXXXX.C from SDBXXXX.C when downloaded to the SUN workstation and proceed with compiling and building the NDB client module. The files NDBCANCR C, NDB_CLNT C, and NDB_XDR C are specifically for the RS/6000. The files NDBCI C, NDBCLNT C, NDB H are for both the RS/6000 and the SUN versions of the NDB client code.

The command for compiling and building the NDB client module is:

```
cc -o ndbcInt ndbci.c ndbcancr.c ndb_clnt.c ndb_xdr.c
```

Chapter 14. Using the Domain Name System

This chapter describes the Domain Name System (DNS), domain name servers, 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 VM DNS includes a name server and a resolver API for application programs. For more information about these services, see “Domain Name Servers” on page 278 and “Resolvers” on page 279. Configuring these services is described in *TCP/IP Planning and Customization*.

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 26 on page 278 is an example of the DNS used in the hierarchy naming structure across an internet.

Using the Domain Name System

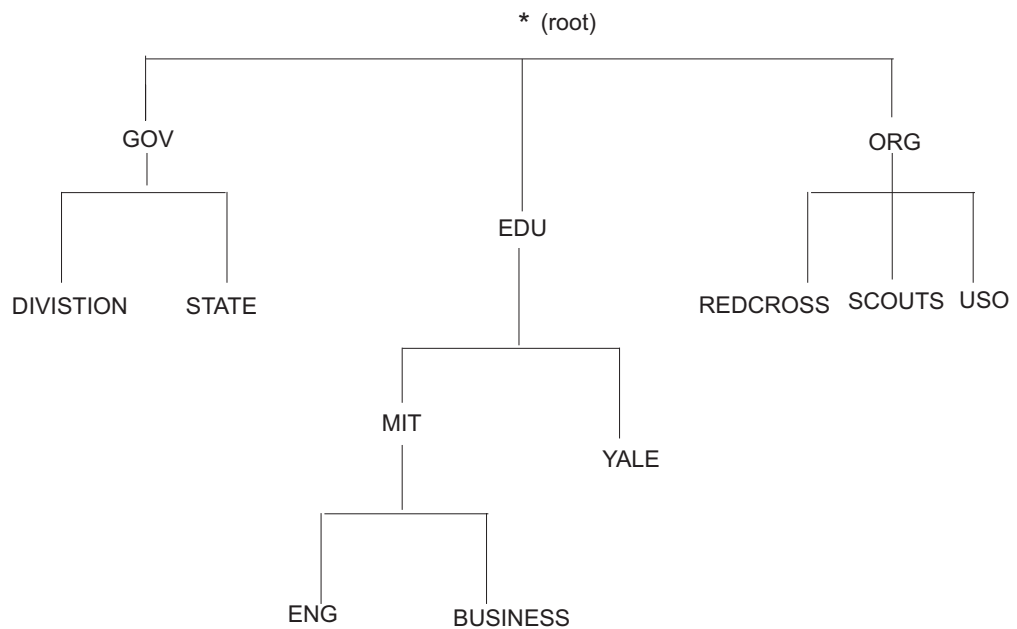


Figure 26. Hierarchical Tree

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 283, “DiG—Querying Name Servers” on page 294, and “CMSRESOL—Resolver and Name Server” on page 305.

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 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 280 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.

Using the Domain Name System

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	<p>Start of authority record</p> <p>The SOA record is unique to a zone. This record contains the administrative details of the zone, including:</p> <ul style="list-style-type: none"> • 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	<p>Name server record</p> <p>The name server record contains the domain name of a name server for the current zone.</p>
A	<p>Address record</p> <p>The address record contains the dotted-decimal notation internet address for the domain name identifying the record.</p>
CNAME	<p>Canonical name record</p> <p>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.</p>
HINFO	<p>Host Information Record</p> <p>This record type contains a text string specifying the CPU (central processing unit) type and operating system of a node.</p>

Using the Domain Name System

MB	<p>The mailbox record (experimental)</p> <p>The mailbox record contains the domain name of a host machine to receive mail for the user specified in the domain name field.</p>
MG	<p>Mail group member record (experimental)</p> <p>The mail group member record specifies the mail address of a person belonging to the mail group specified in the domain name field.</p>
MINFO	<p>Mailbox information record (experimental)</p> <p>The mailbox information record specifies the mail addresses of the persons responsible for the mail group specified in the domain name field.</p>
MR	<p>Mail rename name record (experimental)</p> <p>The mail rename name record specifies a mailbox that is a rename of the mailbox specified in the domain name field.</p>
MX	<p>Mail exchanger record</p> <p>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.</p>
NULL	<p>Null resource record (experimental)</p> <p>The null resource record contains any information, providing it is less than 65 535 octets in length.</p>
PTR	<p>Domain name pointer record</p> <p>The domain name pointer record is mainly used to store data for the <code>in-addr.arpa</code> domain, and contains the domain name referenced by an internet address.</p>
TXT	<p>Text string record</p> <p>The text string record contains descriptive text.</p>
WKS	<p>Well-known services record</p> <p>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.</p>

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

Note: Currently, NSLOOKUP supports IPv4 only.

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 options in command line mode
4. NSLOOKUP options in interactive session mode

NSLOOKUP retains the last value set by any one of the items in the list. Thus, you can configure the internal state information of NSLOOKUP according to the last value set by any one of the items.

For information about the TCPIP DATA file, see *TCP/IP Planning and Customization*.

After parsing the command line, NSLOOKUP attempts to read the startup file, NSLOOKUP ENV. This file contains only the NSLOOKUP options, which define the NSLOOKUP defaults. Enter each option on a separate line; blank lines are not accepted. For more information about the valid options available in NSLOOKUP, see “Set Options” on page 288.

The following is an example of the contents of the NSLOOKUP ENV file:

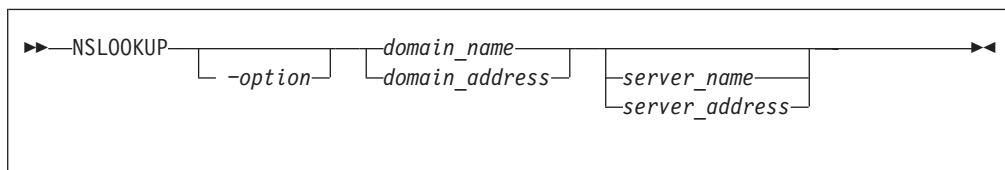
```
set domain=powers.oz
querytype=HINFO
set norecurse
vc
```

Note: Specifying set before the NSLOOKUP options in the NSLOOKUP ENV file is optional.

NSLOOKUP Commands

Use the NSLOOKUP command to either issue multiple name server queries in interactive session mode, or specify an individual query in command line mode.

NSLOOKUP Command Line Query



Note: Parameters and subcommands of NSLOOKUP are case sensitive and must be entered in lowercase. Parameter values and domain names are not case sensitive.

Purpose

Use the NSLOOKUP command to issue a query in command line mode. In command line mode, individual queries are made by specifying a domain name or address on the command line.

Operands

option

Specifies an NSLOOKUP option that tailors query output.

For the available options, see “Set Options” on page 288.

domain_name

Queries the name server for information about the current query type of *domain_name*. The default query type is A (address query).

domain_address

SReverses the components of the address, and generates a pointer type (PTR) query to the name server for the `in-addr.arpa` domain mapping of the address to a domain name.

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.

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` domain is initially made to the default name server to map the internet address to a domain name for the server.

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

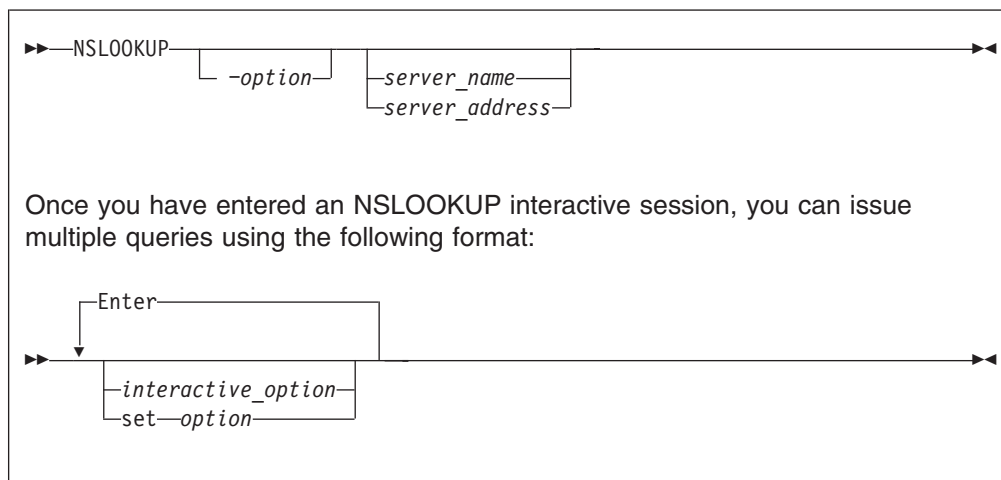
When you specify NSLOOKUP options in command line mode, do not use `set` preceding the option. For example, to specify a name server (NS) type record lookup for the domain name `fourex.oz`, enter the following on the command line:

```
nslookup -querytype=ns fourex.oz
```


The `-querytype=ns` option is a SET subcommand parameter, but `set` is omitted from the command.

For more information about using these options, see “Set Options” on page 288.

NSLOOKUP Interactive Session Queries



Note: Parameters and subcommands of NSLOOKUP are case sensitive and must be entered in lowercase. Parameter values and domain names are not case sensitive.

Purpose

Use the NSLOOKUP command to issue a query in command line mode. In command line mode, individual queries are made by specifying a domain name or address on the command line.

Operands

option

Specifies an NSLOOKUP option that tailors query output.

For the available options, see “Set Options” on page 288.

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

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` domain is initially made to the default name server to map the internet address to a domain name for the server.

interactive_option

Specifies an interactive query option.

set_option

Specifies the set option.

Interactive Options

server {*name*address}

Changes the current server. If *name* is specified, the internet address of *name* is determined using the current server.

An error occurs if the domain name cannot be mapped to an internet address. This option does not ensure that you can contact a name server at the address; it simply changes a local variable storing the address of the default name server.

lserver {*name*address}

Changes the current server. If *name* is specified, the internet address of *name* is determined using the initial server defined at command invocation.

An error occurs if the domain name cannot be mapped to an internet address. This option does not ensure that you can contact a name server at the node specified, it simply changes a local variable storing the address of the default name server.

root

Changes the current server address to the address of the root server. The root server is ns.nic.ddn.mil by default, but can be changed using the *root=name* set subcommand. This command is equivalent to *lserver name*.

An error occurs if the name of the root server cannot be mapped to an internet address. This option does not ensure that you can contact a name server at the node specified; it simply changes a local variable storing the address of the default name server.

finger [*loginname*] [{>|>>} *file_name*]

Extracts information from the finger server of the node found in the last address query. By default, this command returns a list of logged in users for the node last found. You can find information about a particular user by specifying the login name of the user as a parameter. The *loginname* variable is case sensitive and must be specified in the same case (upper or lower) as that used by the host.

You can place output in a file for later viewing by specifying *file_name*. The *> file_name* option places the output in *file_name* overwriting the contents, if any, of the file. The *>> file_name* option places the output in *file_name* appending it to the contents, if any, of the file. There must be at least one space before and after the *>* or *>>* symbol.

An error occurs if the preceding subcommand was not a successful address query or finger operation. If the current host is not defined, querying the name server defines that name server to be the current host for a subsequent finger operation.

The finger option expects that the finger server is operating on the node found. An error occurs if the server is not operating or the node cannot be reached.

ls [-{*aldhlmslt* [*type*]}] *domain* [{>|>>} *file_name*]

Lists the information available for the domain. By default, the internet address of each node in the domain is listed.

To select resource records other than the default, specify one of the following options:

- a** CNAME
- d** ALL

-h HINFO

-m MX

-s WKS

-t [*type*]

Retrieves the resource record type specified in *type*. If no record type is specified with the **-t** option, the current default type is used.

See the resource record field, “Data type” on page 281, for information about valid query types.

The **ls** command expects the domain name specified in *domain* to be a zone. If the domain name specified refers to a host, an error message is printed and no information is given. This command should create a virtual circuit (TCP connection) with the current name server to service the request. An error message is printed if the virtual circuit cannot be established.

Output can be placed in a file for later viewing by specifying *file_name*. The **>** *file_name* option places the output in *file_name* overwriting the contents, if any, of the file. The **>>** *file_name* option places the output in *file_name* appending it to the contents, if any, of the file. There must be at least one space before and after the **>** or **>>** symbol.

A number sign (#) is displayed at the terminal as every 50 lines are written to the file to indicate the command is still executing.

Note: The **ls** command works correctly only if the name server is configured using the one-answer zone transfer format.

view *file_name*

Sorts and lists the contents of *file_name* one screen at a time. An error occurs if the file does not exist.

help or ?

Displays a brief summary of commands.

exit

Exits from NSLOOKUP interactive session mode.

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 **server** or **lserver** options.

You can make a query by entering the domain name of the node or subnetwork for which information is required. Define the data type of information to be retrieved using the **set querytype=** option. 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. If an internet address is given rather than a domain name, a query for the address in the **in-addr.arpa** domain is made to map the internet address to a domain name.

The domain name or address for the 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, typing:

```
toolah wurrup.fourex.oz
```

NSLOOKUP Commands

queries the name server on `wurrrup.fourex.oz` for information about the node `toolah`. 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.

The 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=` option, with the initial default obtained from the TCPIP DATA file. When the `defname` flag is enabled using the `set defname` option, 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 the `defname` flag is enabled, a query for the name `toolah` automatically generates a query packet containing the domain name `toolah.fourex.oz`.

A time-out error occurs if the name server is not running or is unreachable. A Non-existent Domain error occurs if any resource record type for the specified domain name is not available at the name server. A Server Failed error occurs when the local name server cannot communicate with the remote name server.

NSLOOKUP can interpret typing or syntax errors in subcommands as queries. This results in a query being sent and the name server response printed. The response is usually Non-existent Domain, which indicates that the server could not find a match for the query.

Set Options

all Allows you to print the current values of the internal state variables. This option does not alter the internal state of NSLOOKUP.

class=class

Sets the class of information returned by queries. The class must be identified by its mnemonic. For more information about the classes recognized by NSLOOKUP, see the resource record field, "Network class" on page 280. The minimum abbreviation for this option is `c1`.

[no]brackets

Displays output using '<,' and '>,' for terminals that do not support square brackets. The default is brackets.

[no]d2

Directs NSLOOKUP to enable or disable extra debugging mode. Using `d2` also enables debug mode. The default is `nod2`.

[no]debug

Directs NSLOOKUP whether to print debugging information for each query and its corresponding response. The default is `nodebug` (do not print debugging information). Specifying `nodebug` also disables `d2` (extra debugging). The minimum abbreviations for these options are `deb` and `nodeb`.

[no]defname

Specifies whether to append the default domain name to an unqualified domain name in a query.

The default domain name is initially obtained from the TCPIP DATA file, but can be changed using the `domain=name` option.

If the `nodename` option is set, the domain name specified in the query is passed to the server without modification. The default is `defname`. The minimum abbreviations for these options are `def` and `nodef`.

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. This option also updates the search list to contain only the domain name specified. The minimum abbreviation for this option is `do`.

[no]ignoretc

Directs NSLOOKUP on the handling of truncated responses. The name server indicates, in the response header, that the complete query response did not fit into a single UDP packet and has been truncated.

Specifying `ignoretc` directs NSLOOKUP to ignore the truncation condition when it is set in the response by the name server. Specifying `noignoretc` directs NSLOOKUP to automatically retry the query using a TCP connection when a response is sent with the truncation indicator set.

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 only revealed when the debug option is enabled. The default is `ignoretc`. The minimum abbreviations for these options are `ig` and `noig`.

port=*port*

Specifies the port number to use when contacting the name server. The DNS is a well-known service and has been allocated port 53. NSLOOKUP uses port 53 by default, but the port option allows you to specify another port to access. The minimum abbreviation for this option is `po`.

querytype=*type*

Specifies the type of information returned by queries. The initial query type is A (address information). See the resource record field, "Data type" on page 281, for the available query types.

NSLOOKUP cannot generate queries about type NULL. However, it can accept responses containing resource records of type NULL. In this case, NSLOOKUP displays the number of bytes returned in the NULL record. Global queries that return all resource records for a specific domain name are specified by the wildcard value ANY. The minimum abbreviation for this option is `q`.

The `type=type` option is accepted by NSLOOKUP as a synonym for the `querytype=type` option.

[no]recurse

Specifies whether to request a recursive query when querying a name server. The `norecurse` option specifies that a recursive query is not returned. The default is `recurse`. The minimum abbreviations for these options are `rec` and `norec`.

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 NSLOOKUP from contacting the name server. The result is the following error message: no response from server.

NSLOOKUP Commands

The retry algorithm for NSLOOKUP 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. The minimum abbreviation for this option is *ret*.

root=*name*

Specifies the name of a root server. The root server is *ns.nic.ddn.mil* by default.

[no]search

Directs NSLOOKUP to enable or disable the use of a search list. The minimum abbreviations for these options are *sea* and *nosea*.

srchlist=*domain*[/*domain*/...]

Specifies one or up to three domain names to be appended to unqualified host names when attempting to resolve the host name. Each domain name specified is tried in turn until a match is found.

This option also directs the default domain to be set to the first domain name specified in the search list. The minimum abbreviation for this option is *srchl*.

timeout=*interval*

Specifies the number of seconds to wait before timing out of a request. The default for *interval* is retrieved from the TCPIP DATA file. The minimum abbreviation for this option is *t*.

[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).

Internal state information affects the operation and results of your queries. You can change the internal state information maintained by NSLOOKUP using the NSLOOKUP options. Some internal state information is initially retrieved from the TCPIP DATA file. See *TCP/IP Planning and Customization* for more information about the TCPIP DATA file.

The NSLOOKUP options consist of the SET subcommand and its associated parameters. These options can be specified in command line mode queries, interactive session mode queries, or in the NSLOOKUP ENV file. When NSLOOKUP options are specified in command line mode queries, you must omit the word *set* preceding the option. If the NSLOOKUP options are specified in interactive session mode queries, the word *set* must precede the option. In the NSLOOKUP ENV file, specifying the word *set* is optional.

If the NSLOOKUP ENV file exists, the NSLOOKUP options 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 283.

Usage Notes

1. Directs NSLOOKUP to perform a query for the domain nominated. 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.
2. Output can be placed in a file for later viewing by specifying *file_name*. The *> file_name* option places the output in *file_name* overwriting the contents, if any,

of the file. The `>> file_name` option places the output in `file_name` appending it to the contents, if any, of the file. There must be at least one space before and after the `>` or `>>` symbol.

3. 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` 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.
4. Text that does not conform to the defined options and follows the preceding syntax is treated as a domain query. 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.

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 27, 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`.

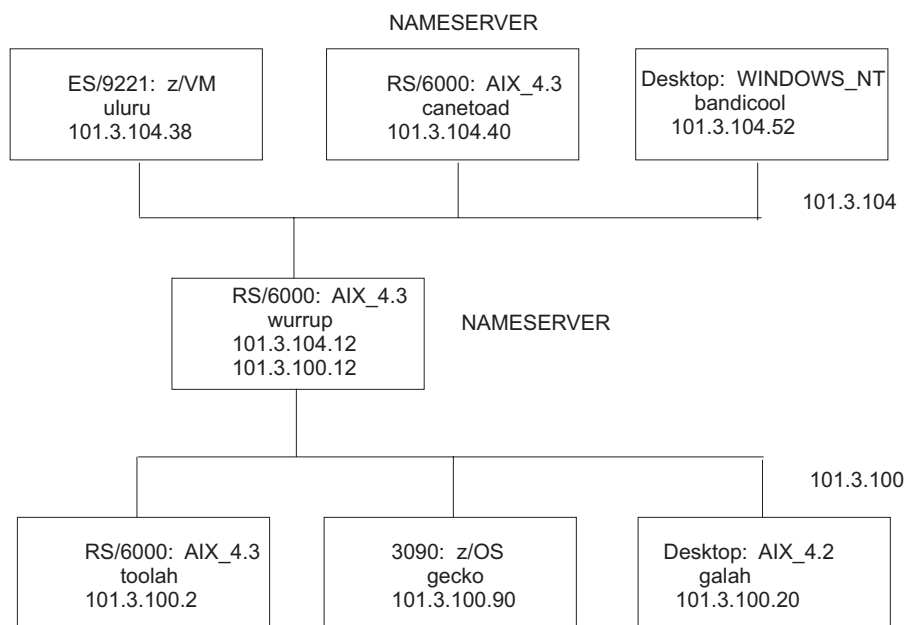


Figure 27. A TCP/IP Network

The following are examples of how to use NSLOOKUP to extract information from a name server. The queries are executed from the VM host `uluru` at IP address `101.3.104.38` on the network described in Figure 27.

The following examples are command line mode queries.

1. To make a simple address query:

NSLOOKUP Commands

```
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
```



```
wurrrup.fourex.oz  internet address = 101.3.100.12
wurrrup.fourex.oz  internet address = 101.3.104.12
canetoad.fourex.oz internet address = 101.3.104.40
```

5. Change the current server from wurrrup 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
       wurrrup
System: >; >; Server:  canetoad.FOUREX.OZ
Address: 101.3.104.40
```

```
res_mkquery(0, wurrrup.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:
    wurrrup.FOUREX.OZ, type = A, class = IN
  ANSWERS:
    ->; wurrrup.FOUREX.OZ
      internet address = 101.3.104.12
      ttl = 9999999 (115 days 17 hours 46 mins 39 secs)
    ->; wurrrup.FOUREX.OZ
      internet address = 101.3.100.12
      ttl = 9999999 (115 days 17 hours 46 mins 39 secs)
```

```
-----
Name:  wurrrup.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 option:

```
User:  ls fourex.oz
System: >; canetoad.FOUREX.OZ

fourex.oz          server = wurrrup.fourex.oz
wurrrup            101.3.100.12
wurrrup            101.3.104.12
fourex.oz          server = canetoad.fourex.oz
canetoad           101.3.104.40
gecko              101.3.100.90
wurrrup            101.3.100.12
wurrrup            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
```

NSLOOKUP Commands

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

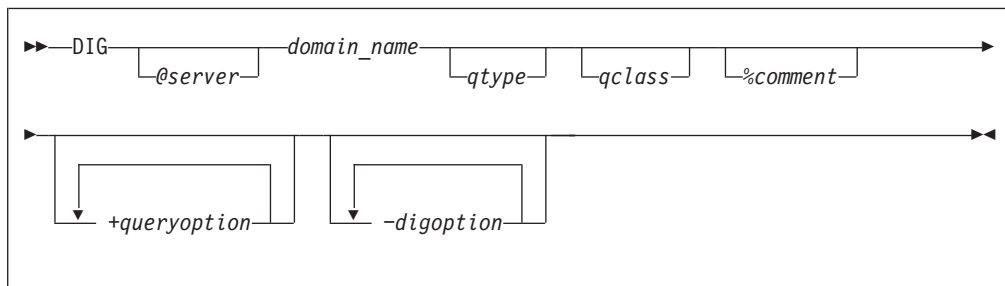
The internal state information of DIG determines the operation and results of your name server queries. You can configure the internal state information of DIG using the following methods, listed in order of precedence:

- 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 296.

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.

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” on page 281.

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” on page 280.

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 296 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 298 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 `ndefname` 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.

DIG Command

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. 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 on page 295, 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 294.

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 294.

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
wurrrup 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 294.

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 on page 295 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 *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 27 on page 291, 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 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:
```



```
wurrrup.fourex.oz A      101.3.100.12
wurrrup.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      wurrrup.fourex.oz adb.wurrrup.fourex.oz (
                                10003 ;serial
                                3600  ;refresh
                                300   ;retry
                                3600000 ;expire
                                86400 ) ;minim
```

4. Use the default query options:

```
System: Ready
User: dig wurrrup
System: ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; ANSWERS:
wurrrup.FOUREX.OZ. A      101.3.104.12
wurrrup.FOUREX.OZ. A      101.3.100.12
```

5. Print the network class information:

```
System: Ready
User: dig wurrrup +cl
System: ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; ANSWERS:
wurrrup.FOUREX.OZ. IN A    101.3.104.12
wurrrup.FOUREX.OZ. IN A    101.3.100.12
```

6. Echo the input query:

```
System: Ready
User: dig wurrrup +cmd
System: ; <<>> DiG 2.0 <<>> wurrrup +cmd
; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; ANSWERS:
wurrrup.FOUREX.OZ. A      101.3.104.12
wurrrup.FOUREX.OZ. A      101.3.100.12
```

7. Print the question section of the output:

```
System: Ready
User: dig wurrrup +qu
System: ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; QUESTIONS:
;;      wurrrup.FOUREX.OZ, type = A, class = IN

;; ANSWERS:
wurrrup.FOUREX.OZ. A      101.3.104.12
wurrrup.FOUREX.OZ. A      101.3.100.12
```

8. Do not print the header line:

```
System: Ready
User: dig wurrrup +noH
System: ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; ANSWERS:
wurrrup.FOUREX.OZ. A      101.3.104.12
wurrrup.FOUREX.OZ. A      101.3.100.12
```

9. Print the query flags:

```
System: Ready
User: dig wurrrup +he
System: ;; flags: qr aa rd ra ; Ques: 1, Ans: 2, Auth: 0, Addit:
;; ANSWERS:
wurrrup.FOUREX.OZ. A      101.3.104.12
wurrrup.FOUREX.OZ. A      101.3.100.12
```

10. Print the question section and the outgoing query:

```
System: Ready
User: dig wurrrup +qu +qr
System: ; Ques: 1, Ans: 0, Auth: 0, Addit: 0
      ;; QUESTIONS:
      ;;      wurrrup.FOUREX.OZ, type = A, class = IN

      ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
      ;; QUESTIONS:
      ;;      wurrrup.FOUREX.OZ, type = A, class = IN

      ;; ANSWERS:
      wurrrup.FOUREX.OZ. A      101.3.104.12
      wurrrup.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      wurrrup.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      wurrrup.fourex.oz
      fourex.oz 9999999 NS      canetoad.fourex.oz
```

13. Enable extra debugging mode:

```
System: Ready
User: dig wurrrup +d2
System: ;; res_mkquery(0, wurrrup, 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:
      wurrrup.FOUREX.OZ. A      101.3.104.12
      wurrrup.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 wurrrup +nodefname +qu
System: ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
      ;; QUESTIONS:
      ;;      wurrrup, type = A, class = IN

      ;; ANSWERS:
      wurrrup.fourex.oz A      101.3.104.12
      wurrrup.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 wurrrup +do=fourexpd +qu
System: ;; -->HEADER<<- opcode: QUERY , status: SERVFAIL, id: 3
```

```

; Ques: 1, Ans: 0, Auth: 0, Addit: 0
;; QUESTIONS:
;;      wurrrup.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      wurrrup.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 wurrrup -c any +qu
System:  ; Ques: 1, Ans: 2, Auth: 0, Addit: 0
;; QUESTIONS:
;;      wurrrup.FOUREX.OZ, type = A, class = ANY

;; ANSWERS:
wurrrup.FOUREX.OZ.  A      101.3.104.12
wurrrup.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 wurrrup -t any +qu
System:  ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
;; QUESTIONS:
;;      wurrrup.FOUREX.OZ, type = ANY, class = IN

;; ANSWERS:
wurrrup.FOUREX.OZ.  A      101.3.104.12
wurrrup.FOUREX.OZ.  A      101.3.100.12
wurrrup.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.

```

wurrrup any in +noH +nohe +noqu +noad +noau -envset -stick
wurrrup 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 wurrrup any in +noH +nohe +noqu +noad
        +noau -envset -st k
        ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
        ;; ANSWERS:
        wurrrup.FOUREX.OZ. 9999999 A      101.3.104.12
        wurrrup.FOUREX.OZ. 9999999 A      101.3.100.12
        wurrrup.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 wurrrup any in
        ; Ques: 1, Ans: 3, Auth: 0, Addit: 0
        ;; ANSWERS:
        wurrrup.FOUREX.OZ. 9999999 A      101.3.104.12
        wurrrup.FOUREX.OZ. 9999999 A      101.3.100.12
        wurrrup.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: 404612488 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 DB2 Server for VM database, or other name servers as its source of information.

Note: Currently, CMSRESOL supports IPv4 only.

RESOLV Command—Interface to the CMSRESOL MODULE

```

>>—RESOLV—quest_name—quest_type—nsaddr—>>

```

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).

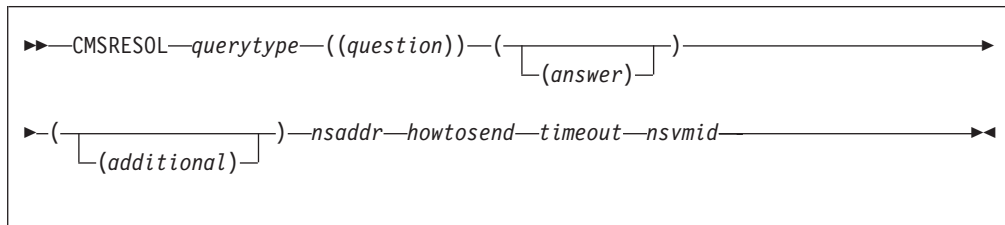
The valid values for this parameter are listed in the header file CMOLVER.H. They are also listed in RFC 883.

CMSRESOL Command

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, UDP, or CMS IUCV. 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, IQUERY, or DBASE.

QUERY Specifies the standard query

IQUERY Specifies the inverse query

DBASE Specifies the database query or update.

question

Specifies the form of Qname, Qtype, and Qclass.

answer

Specifies the form of Name, Type, Class, TTL, and Data.

additional

Used for database queries and updates.

See “The Database Query and Update” on page 308 for the syntax of database queries and updates.

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

IUCV Specifies the use of CMS IUCV.

timeout

Indicates the number of seconds to wait when attempting to open a connection to the name server.

nsvmid

Specifies the VM user ID of the virtual machine that is running the name server. This is required for IUCV processing only.

The *question*, *answer*, *authority*, and *additional* 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

Notes:

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 ) ) ( ) ( ) 127.0.0.1
                DATAGRAM 60 NAMESRV
```

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 ) ) ( )
                  127.0.0.1 CONN 60 NAMESRV
```

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 ) ) ( ) ( ) 127.0.0.1
                DATAGRAM 45 NAMESRV
```

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 ) )
( )
( )
```

The Database Query and Update

A database query and update permits a client to update the DB2 Server for VM database. The database query and update functions are enhancements provided by IBM. An authorization scheme protects the database from unauthorized updates. The name field of a resource record (RR) type 97 defines a domain name; the data field defines a list of VM user IDs that can perform a database update. The serial value specified on the start of authority (SOA) record is not updated to reflect a change in the zone data. Other locations that transfer the domain do not do so until the serial on the SOA record is modified, or the refresh timer expires. See RFC 1034 or 1035 for further information on SOA records.

An example of a database update follows. IUCV is required for database updates. UDP datagrams or TCP virtual circuits are not permitted.

```
CMSRESOL DBASE ( ) ( ) ( ( RALPH.YKT.IBM.COM UPDATE IN 0 #A RSRVD
                        This is the NewData# ) ) 127.0.0.1 IUCV 60 NAMESRV
```

The stack contents are:

```
( )
( )
( )
( )
```

Note: In a database update, no data is returned between the parentheses.

The database functions use the *additional* sections to transport the request to the name server. Figure 28 on page 309 provides the format of the data within the *additional* section:

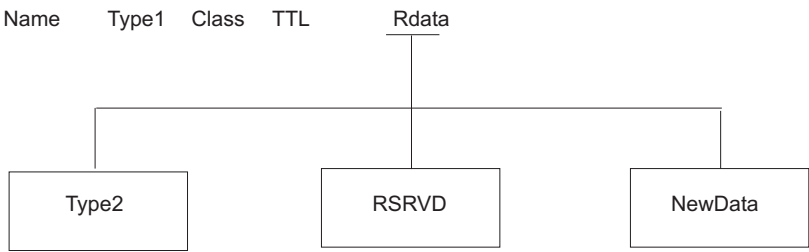


Figure 28. The Database Format of an Additional Section

Type1 can be update or select for database functions. The Rdata field is surrounded by the # character. Within the Rdata field, there are three subfields for update operations: Type2, RSRVD, and NewData. This example uses an A-type internet address, which for the IN (Internet system) class is a 32-bit internet protocol address.

Subfield	Description
Type2	Specifies the record type to be updated
RSRVD	Is a reserve field
NewData	Is the new data

For database updates, a TYPE97 resource record (RR) is examined for authorization checking. For more information, see *TCP/IP Planning and Customization*.

The following is an example of a database query using UDP datagrams. For a query operation, TCP virtual circuits, UDP datagrams, or CMS IUCV are allowed.

```
CMSRESOL DBASE ( ) ( ) ( ( RALPH.YKT.IBM.COM SELECT IN 0 A ) )
127.0.0.1 DATAGRAM 60 NAMESRV
```

The stack contents are:

```
( )
( ( RALPH.YKT.IBM.COM A IN 9999999 192.5.5.5 ) )
( )
( ( RALPH.YKT.IBM.COM SELECT IN 0 A ) )
```

The database query operation, like the update operation, uses the *additional* section to transport the request to the name server. The type in this case is SELECT, indicating a database query operation. The Rdata field indicates the type of RR to query. In this example, the type of RR to query is the internet address Type A. The results are returned in the *answer* section.

Querying Outside Your Domain

The following is a sample of a query outside your domain:

```
CMSRESOL QUERY ( ( ALPHA.UNKNOWN.COM A IN ) ) ( ) ( )
127.0.0.1 DATAGRAM 30 NAMESRV
```

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 ) )
```

Types of Queries

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 21 on page 314. 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.

Using Translation Tables

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 320.

Table 19. 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 TCPxxBIN 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 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 TCPxxBIN, if it is available, or from an equivalent that is compiled into the program.

Table 20 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 20. 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	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
LPD Server	TRANSLATETABLE table_name ²	LPD
NFS Server	XLATE=table_name ²	VMNFS
TELNET Client	TRANSLATE table_name ²	TELNET
TELNET Server (line mode)	None	STLINMOD
TFTP Client	TRANSLATE table_name ²	TFTP
Note¹: For SMTP, an additional translation table may be specified for 8-bit MIME support. The <i>TCP/IP Planning and Customization</i> contains more information in the “8BITMIME Statement” section.		
Note²: table_name is the file name of a TCP/IP translation table.		
Note³: This applies only when the MIME option is specified on the SENDFILE command.		

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 *TCP/IP User's Guide* contains information on the TRANSLATE option for the TCP/IP clients, and the *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.

Using Translation Tables

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 320.

Table 21. IBM Translation Tables

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
United States and Canada	0037rrrr	TCPXLATE	37	rrrr
Austria and Germany	0273rrrr	TCPXLATE	273	rrrr
Denmark and Norway	0277rrrr	TCPXLATE	277	rrrr
Finland and Sweden	0278rrrr	TCPXLATE	278	rrrr
Italy	0280rrrr	TCPXLATE	280	rrrr
Spain and Spanish-speaking Latin America	0284rrrr	TCPXLATE	284	rrrr
United Kingdom	0285rrrr	TCPXLATE	285	rrrr
France	0297rrrr	TCPXLATE	297	rrrr
International	0500rrrr	TCPXLATE	500	rrrr
Iceland	0871rrrr	TCPXLATE	871	rrrr
ISO 8859-15	0924rrrr	TCPXLATE	924	rrrr
OpenExtensions (POSIX)	1047rrrr	TCPXLATE	1047	rrrr
United States and Canada (Euro)	1140rrrr	TCPXLATE	1140	rrrr
Austria and Germany (Euro)	1141rrrr	TCPXLATE	1141	rrrr
Denmark and Norway (Euro)	1142rrrr	TCPXLATE	1142	rrrr
Finland and Sweden (Euro)	1143rrrr	TCPXLATE	1143	rrrr
Italy (Euro)	1144rrrr	TCPXLATE	1144	rrrr
Spain and Spanish-speaking Latin America (Euro)	1145rrrr	TCPXLATE	1145	rrrr
United Kingdom (Euro)	1146rrrr	TCPXLATE	1146	rrrr
France (Euro)	1147rrrr	TCPXLATE	1147	rrrr
International (Euro)	1148rrrr	TCPXLATE	1148	rrrr

Table 21. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
Iceland (Euro)	1149rrrr	TCPXLATE	1149	rrrr
OpenExtensions	1047rrrr	TCPXLATE	1047	rrrr
ISO 8859-15 (EBCDIC)	0924rrrr	TCPXLATE	924	rrrr
ISO 8859-15 (ASCII)	hhhh0923	TCPXLATE	hhhh	923
OS/2	hhhh0850	TCPXLATE	hhhh	850
OS/2 (Euro)	hhhh0858	TCPXLATE	hhhh	858
ISO 8859-1 (ASCII)	hhhh0819	TCPXLATE	hhhh	819
Microsoft® Windows®	hhhh1252	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		
			300	X0208
JIS X0208 1978			300	X0208
JIS X0208 1983			300	X0208
Shift JIS X0208			300	EUC
Extended Unix Code IBM			300	300
Standard Korean Hangeul	STANDARD	TCPHGLAT		
			833	1088
KSC 5601 SBCS			834	951
KSC 5601 DBCS			833	891
Hangeul SBCS			834	926
Hangeul DBCS				

Using Translation Tables

Table 21. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
Standard Traditional Chinese	STANDARD	TCPCHLAT		
			037	904
Traditional Chinese SBCS			835	927
Traditional Chinese DBCS				

Note: In this table *hhhh* and *rrrr* represent four-digit host and remote code page numbers, respectively.

Notes:

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 1047rrrr, 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
;
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
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

Using Translation Tables

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 ..
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 ..
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

```

Using Translation Tables

```
09 FF      09 05      09 FF      09 05
:
```

Traditional Chinese DBCS Translation Tables

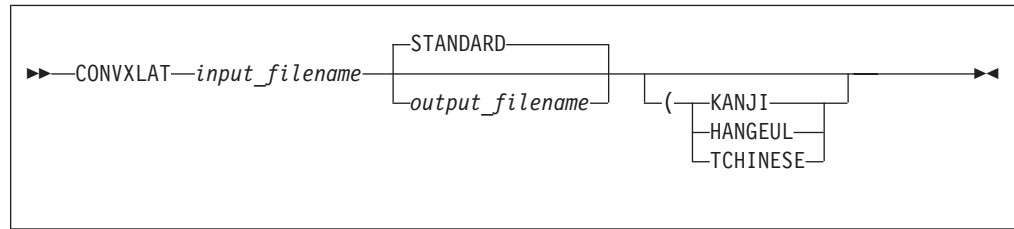
```
;
; STANDARD TCPCHLAT - Traditional Chinese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPCHBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID 927
; TCHETA      TCHATE
;
4040 8140      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.

▶▶—/directory_name/file_name————▶◀

Parameter	Description
<i>directory_name</i>	Specifies a directory name. Directories contain the names of files, other directories, or both.
<i>file_name</i>	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:

<i>mailfiles</i>	Is the directory name.
<i>cooks</i>	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*.

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 98 Files

DOS, OS/2, and Windows 98 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 98 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 98, 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 98 supports 256-character filenames as well as upper and lowercase characters.

Specifying Files and Data Sets

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 98 file names that are valid.

```
TEMP
START.BAT
A:COMMAND.COM
C:SPF\SPFPC.HLP
C:REPORTS\ThisISALongFilename.TXT    (Windows 98)
```

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.

►►—*library/file.member*—————►◄

Parameter	Description
<i>library</i>	Is a library name. Libraries contain the names of programs, files, and commands.
<i>file.member</i>	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
```

Several sample NETRC DATA file entries follow:

```
machine oddjob login anonymou password anonymou
machine 123.45.67.89 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
```

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 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

NETRC DATA File

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 command, 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 22 lists the mapping values for the APL2 character set.

Table 22. 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

Mapping APL2 Character Set Values

Table 22. 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

Mapping APL2 Character Set Values

Table 22. 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

Mapping APL2 Character Set Values

Table 22. 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

Mapping APL2 Character Set Values

Table 22. 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

Mapping APL2 Character Set Values

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.

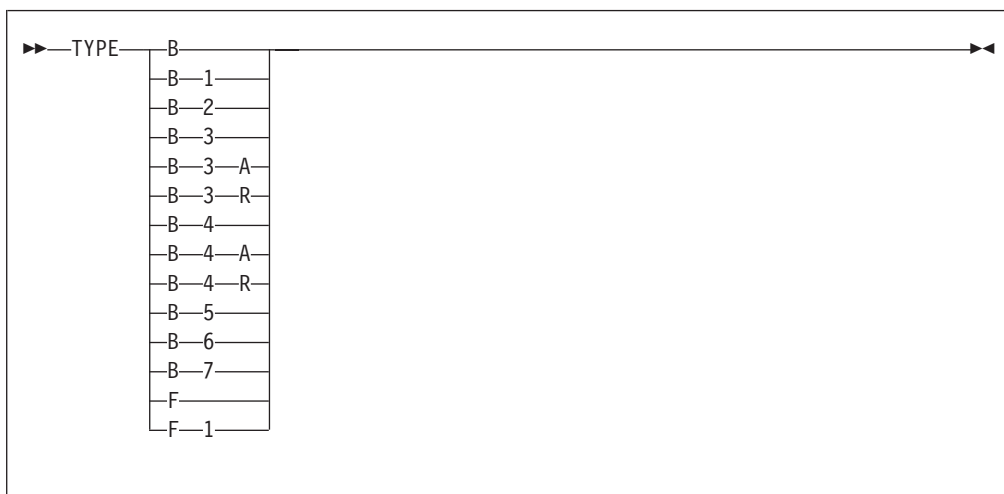
DBCS Subcommands

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

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 23 indicates the actual server command that would be generated for each client subcommand alias:

Table 23. 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 *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 one 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), the MIB-II, and the IBM 3172 enterprise-specific variables 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)
- IBM 3172 Enterprise-Specific

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 administrator.

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 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 29 illustrates the hierarchical ISO tree that has been adopted to identify managed objects.

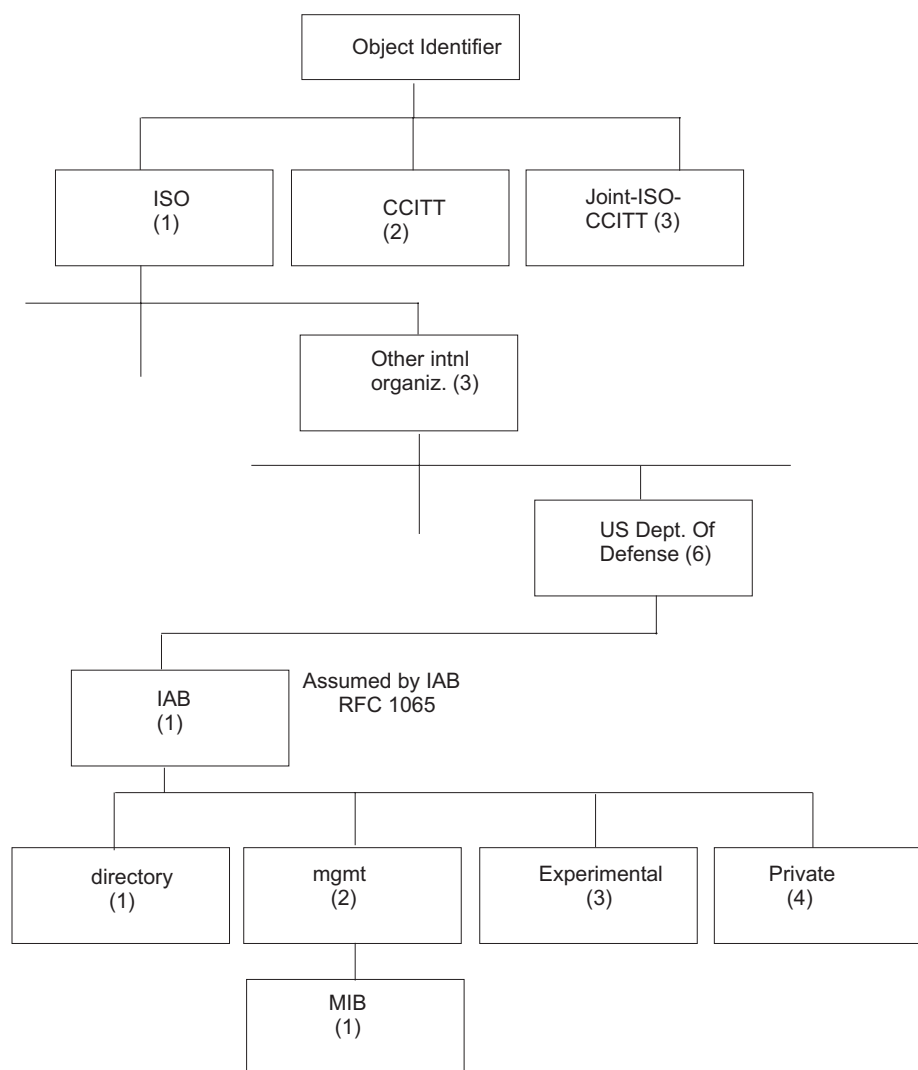


Figure 29. The SMI Hierarchical Tree

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.

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 24 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 24. 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
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 <i>what kind of box</i> is being managed. For example, if vendor <i>Stones, Inc.</i> 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 <i>Fred Router</i> .	read-only
sysUpTime { system 3 }	TimeTicks	The time (in hundredths of a second) since the network management portion of the system was last started.	read-only

1. All accesses of read-write indicate a read-only access for VM.

MIB Objects

Table 24. Implementation of the System Group (continued)

Object	Syntax	Definition	Access										
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, <i>telephone closet, 3rd floor</i>)	read-only										
sysServices { system 7 }	INTEGER	<p>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:</p> <p>Layer Functionality</p> <table><tr><td>1</td><td>Physical (for example, repeaters)</td></tr><tr><td>2</td><td>Datalink/subnetwork (for example, bridges)</td></tr><tr><td>3</td><td>Internet (for example, supports the IP)</td></tr><tr><td>4</td><td>End-to-end (for example, supports the TCP)</td></tr><tr><td>7</td><td>Applications (for example, supports the SMTP)</td></tr></table> <p>For systems including OSI protocols, layers 5 and 6 can also be counted.</p>	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)	read-only
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)												

Interfaces Group

Table 25 lists 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 25. 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

Table 25. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifTable { interfaces 2 }	SEQUENCE of IfEntry	A list of interface entries. The number of entries is given by the value of ifNumber.	not applicable
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

MIB Objects

Table 25. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
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
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), hdlc (3), ddn-x25 (4), rfc877-x25 (5), 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), hdlc (17), tl-carrier (18), cept (19), basicISDN (20), primaryISDN (21), propPointToPointSerial (22), terminalServer-asynCPort (23), softwareLoopback (24), eoc (25), ethernet-3Mbit (26), nsip (27), slip (28)	The type of interface, distinguished according to the physical/link/network protocol(s) immediately <i>below</i> 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

Table 25. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
ifPhysAddress { ifEntry 6 }	OCTET STRING	The interface's address at the protocol layer immediately <i>below</i> 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
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.	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

MIB Objects

Table 25. Implementation of the Interfaces Group (continued)

Object	Syntax	Definition	Access
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
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	<p>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:</p> <pre>nullSpecific OBJECT IDENTIFIER ::= { 0 0 }</pre> <p>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.</p>	read-only

Address Translation Group

Table 26 on page 351 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 26. 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 (for example, DDN-X.25 has an algorithmic method). 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 27 lists the objects in the IP group. The IP objects are the statistics and gateway routing tables for the IP layer.

Table 27. 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
ipInReceives { ip 3 }	Counter	The number of input datagrams received from interfaces, including those received in error.	read-only

MIB Objects

Table 27. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
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
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

Table 27. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
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 <i>Don't Fragment</i> 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, ipAdEntIfIndex 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
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

MIB Objects

Table 27. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
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
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

Table 27. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
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
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), bbnSpflgp (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

MIB Objects

Table 27. Implementation of the IP Group (continued)

Object	Syntax	Definition	Access
ipRouteAge { ipRouteEntry 10 }	INTEGER	The number of seconds since this route was last updated or otherwise determined to be correct. Note semantics of <i>too old</i> cannot be implied, except through knowledge of the routing protocol by which the route was learned.	read-write
ipRouteMask { ipRouteEntry 11 }	ipAddress	<p>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:</p> <p>mask network 255.0.0.0 class-A 255.255.0.0 class-B 255.255.255.0 class-C</p> <p>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.</p>	read-write

IP Address Translation Table

Table 28 lists the objects in the IP Translation table.

Table 28. 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
IpNetToMediaEntry { ipNetToMediaTable 1 }	<p>IpNetToMediaEntry ::= SEQUENCE</p> <p>ipNetToMediaIfIndex INTEGER, ipNetToMediaPhysAddress OCTET STRING, ipNetToMediaNetAddress IpAddress, ipNetToMediaType INTEGER</p>	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

Table 28. IP Address Translation Table (continued)

Object	Syntax	Definition	Access
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 29 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 29. 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
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

MIB Objects

Table 29. Implementation of the ICMP Group (continued)

Object	Syntax	Definition	Access
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
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

Table 29. Implementation of the ICMP Group (continued)

Object	Syntax	Definition	Access
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 30 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 30. 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
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

MIB Objects

Table 30. Implementation of the TCP Group (continued)

Object	Syntax	Definition	Access
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
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 connection-specific information.	read-only

Table 30. Implementation of the TCP Group (continued)

Object	Syntax	Definition	Access
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 31 lists the objects in the UDP group. The UDP objects are the datagram statistics of the UDP layer.

Table 31. Implementation of the UDP Group

Object	Syntax	Definition	Access
udpInDatagrams { udp 1 }	Counter	The number of UDP datagrams delivered to UDP users.	read-only

MIB Objects

Table 31. Implementation of the UDP Group (continued)

Object	Syntax	Definition	Access
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

IBM 3172 Enterprise-Specific MIB Variables

The following tables list the objects in the various IBM 3172 System tables. These are available if NETMAIN has been specified on the DEVICE statement in the TCP/IP configuration file. See *TCP/IP Planning and Customization* for information about this file.

ibm3172SystemTable

Table 32 on page 363 lists the objects in the IBM 3172 System Table. The system table identifies the hardware, software, contact person, physical location and number of network interfaces for the 3172.

Table 32. Implementation of the IBM 3172 System Table

Object	Syntax	Definition	Access
ibm3172SystemTable	ibm3172SystemTable ::= SEQUENCE	Descriptive objects related to the entire 3172.	read-only
	ibm3172Descr DISPLAYSTRING, ibm3172Contact DISPLAYSTRING, ibm3172Location DISPLAYSTRING, ibm3172ifNumber INTEGER	ASN.1 notation: 1.3.6.1.4.1.2.6.1.1	
ibm3172Descr	DisplayString	Text Description of the 3172. Contains information about the hardware and software of the 3172. The format of the ibm3172Descr variable is : ttttMODELxmmm, xSERIALxNUMBERxsssssssss, xxxxxxxxxxxxl, xPROGRAMxNUMBERxppppppp ; where : x represents a blank character, ; UPPER CASE letters are hardcoded characters, ; (,) represents a comma, ; and the remaining lower case letters represent variable data as follows: t - machine type m - model number s - serial number i - software program name l - software level numbers p - software program product number.	read-only
{ibm3172SystemTable 1}	(SIZE(0..253))	An example of the information sent with this attribute would be: "3172 MODEL 001, SERIAL NUMBER 000001234, 3172 Interconnect Ctlr Program 020100, 5601433"	
ibm3172Contact	DisplayString	The textual identification of the contact person for this 3172, together with information about how to contact this person. This information is provided by the 3172 Operator Facility.	read-only
{ibm3172SystemTable 2}	(SIZE(0..32))		
ibm3172Location	DisplayString	The physical location of this node. This information is provided by the 3172 Operator Facility.	read-only
{ibm3172SystemTable 3}	(SIZE(0..32))		
ibm3172ifNumber	Integer	The number of network interfaces (regardless of their current status) on which this 3172 can send data.	read-only
{ibm3172SystemTable 4}			

ibm3172ifTrapTable

Table 33 on page 364 lists the objects in the IBM 3172 Trap Table. The trap table identifies the trap settings for the interface of the 3172.

MIB Objects

Table 33. Implementation of the IBM 3172 Trap Table

Object	Syntax	Definition	Access
ibm3172ifTrapTable	ibm3172ifTrapTable ::= SEQUENCE	Objects at the interface level pertaining to the trap function.	read-only
	ibm3172ifTrapEnable INTEGER	ASN.1 notation: 1.3.6.1.4.1.2.6.1.2	
ibm3172ifTrapEnable {ibm3172ifTrapTable 1}	Integer	Flag to indicate whether the 3172 should send traps for this interface to the SNMP proxy agent. "0" indicates the trap function of the 3172 is disabled, "1" indicates that it is enabled. The VM SNMP proxy agent does not utilize this function of the 3172, so the value of this variable is always "0".	read-only

ibm3172ifChanCounters Table

Table 34 lists the objects in the IBM 3172 channel counter table. The channel counters identify the inbound and outbound octets and blocks of the 3172.

Table 34. Implementation of the IBM 3172 ChanCounter Table

Object	Syntax	Definition	Access
ibm3172ifChanCounters	ibm3172ifChanCounters ::= SEQUENCE	Objects at the subnetwork layer and below pertaining to a pair of subchannels of the 3172. The values supplied for ibm3172ifOutChanOctets and ibm3172ifOutChanBlocks apply to the outbound subchannel on which the command was received. The values supplied for ibm3172ifInChanOctets and ibm3172ifInChanBlocks apply to the inbound subchannel that is paired with the outbound subchannel on which the command was received.	read-only
	ibm3172ifInChanOctets COUNTER, ibm3172ifOutChanOctets COUNTER, ibm3172ifInChanBlocks COUNTER, ibm3172ifOutChanBlocks COUNTER	ASN.1 notation: 1.3.6.1.4.1.2.6.1.3	
ibm3172ifInChanOctets {ibm3172ifChanCounters 1}	Counter	The number of inbound octets that were transmitted to the host by the 3172, including all headers. The value applies to the inbound subchannel that is paired with the outbound subchannel on which the command was received.	read-only
ibm3172ifOutChanOctets {ibm3172ifChanCounters 2}	Counter	The number of outbound octets that were received from the host by the 3172, including all headers. The value applies to the outbound subchannel on which the command was received.	read-only
ibm3172ifInChanBlocks {ibm3172ifChanCounters 3}	Counter	The number of inbound blocks that were transmitted to the host by the 3172. The value applies to the inbound subchannel that is paired with the outbound subchannel on which the command was received.	read-only

Table 34. Implementation of the IBM 3172 ChanCounter Table (continued)

Object	Syntax	Definition	Access
ibm3172ifOutChanBlocks {ibm3172ifChanCounters 4}	Counter	The number of outbound blocks that were received from the host by the 3172. The value applies to the outbound subchannel on which the command was received.	read-only

ibm3172ifLANCounters Table

Table 35 lists the objects in the IBM 3172 LAN counter table. The LAN counters identify the inbound and outbound octets and frames of the 3172. The LAN counters identify transmission errors, discarded frames, and undeliverable frames.

Table 35. Implementation of the IBM 3172 LAN Counter Table

Object	Syntax	Definition	Access
ibm3172ifLANCounters	ibm3172ifLANCounters ::= SEQUENCE	Objects at the subnetwork layer and below pertaining to a particular LAN of the 3172.	read-only
	ibm3172ifInLANOctets COUNTER, ibm3172ifOutLanOctets COUNTER, ibm3172ifInLANFrames COUNTER, ibm3172ifOutLANFrames COUNTER, ibm3172ifInLANErrors COUNTER, ibm3172ifOutLANErrors COUNTER, ibm3172ifInLANDiscards COUNTER, ibm3172ifOutLANDiscards COUNTER	ASN.1 notation: 1.3.6.1.4.1.2.6.1.4	
ibm3172ifInLANOctets {ibm3172ifLANCounters 1}	Counter	The number of inbound octets that were received from the LAN by the 3172, including all headers.	read-only
ibm3172ifOutLanOctets {ibm3172ifLANCounters 2}	Counter	The number of inbound octets that were transmitted to the LAN by the 3172, including all headers.	read-only
ibm3172ifInLANFrames {ibm3172ifLANCounter 3}	Counter	The number of inbound frames that were received from the LAN by the 3172.	read-only
ibm3172ifOutLANFrames {ibm3172ifLANCounters 4}	Counter	The number of outbound frames that were transmitted to the LAN by the 3172.	read-only
ibm3172ifInLANErrors {ibm3172ifLANCounters 5}	Counter	The number of inbound frames received from the LAN by the 3172 that contained errors preventing them from being deliverable to a higher layer protocol. This variable, when combined with ibm3172inBlkErrors, reflects the total number of inbound frames not forwarded from the LAN to the host because of errors.	read-only

MIB Objects

Table 35. Implementation of the IBM 3172 LAN Counter Table (continued)

Object	Syntax	Definition	Access
ibm3172ifOutLANErrors {ibm3172ifLANCounters 6}	Counter	The number of outbound frames that could not be transmitted to the LAN because of transmission failures. This variable, when combined with ibm3172ifOutDbkErrors, reflects the total number of outbound frames not transmitted to the LAN because of transmission errors.	read-only
ibm3172ifInLANDiscards {ibm3172ifLANCounters 7}	Counter	The number of inbound frames received from the LAN that were discarded by the 3172, even though no errors had been detected to prevent their being deliverable to a higher layer protocol. One possible reason for discarding such a frame could be because of insufficient buffer space. This variable, when combined with ibm3172ifInBlkDiscards, reflects the total number of inbound frames not forwarded from the LAN when no error was detected.	read-only
ibm3172ifOutLANDiscards {ibm3172ifLANCounters 8}	Counter	The number of outbound frames that are discarded.	read-only

ibm3172ifBlkCounters Table

Table 36 lists the objects in the IBM 3172 Blocker counter table. The Blocker counters identify the transmitted and received inbound octets and frames by the LAN adapter. The Blocker counters also identify the tasks that contained errors and the tasks that were discarded.

Table 36. Implementation of the IBM 3172 Blk Counter Table

Object	Syntax	Definition	Access
ibm3172ifBlkCounters	ibm3172ifBlkCounters ::= SEQUENCE ibm3172ifBlkRcvOctets COUNTER, ibm3172ifBlkXmitOctets COUNTER, ibm3172ifBlkRcvFrames COUNTER, ibm3172ifBlkXmitBlocks COUNTER, ibm3172ifInBlkErrors COUNTER, ibm3172ifInBlkDiscards COUNTER	Objects at the Subnetwork layer and below pertaining to a particular Blocker Task of the 3172. ASN.1 notation: 1.3.6.1.4.1.2.6.1.5	read-only
ibm3172ifBlkRcvOctets {ibm3172ifBlkCounters 1}	Counter	The number of inbound octets that were received by the Blocker from the LAN, including all headers.	read-only
ibm3172ifBlkXmitOctets {ibm3172ifBlkCounters 2}	Counter	The number of inbound octets that were transmitted to the channel adapter by the Blocker, including all headers.	read-only

Table 36. Implementation of the IBM 3172 Blk Counter Table (continued)

Object	Syntax	Definition	Access
ibm3172ifBlkRcvFrames {ibm3172ifBlkCounters 3}	Counter	The number of inbound frames that were received from the LAN adapter by the Blocker Task.	read-only
ibm3172ifBlkXmitBlocks {ibm3172ifBlkCounters 4}	Counter	The number of inbound frames that were transmitted to the channel adapter by the blocker task.	read-only
ibm3172ifInBlkErrors {ibm3172ifBlkCounters 5}	Counter	The number of inbound frames transmitted by the LAN adapter to the Blocker Task which contained errors preventing them from being deliverable to a higher layer protocol. This variable, when combined with ibm3172ifInLANErrors, reflects the total number of inbound frames discarded by the 3172 because of errors.	read-only
ibm3172ifInBlkDiscards {ibm3172ifBlkCounters 6}	Counter	The number of inbound frames transmitted by the LAN adapter to the Blocker Task that were discarded by the 3172, even though no errors had been detected to prevent their being deliverable to a higher layer protocol. One possible reason for discarding such a frame could be because of insufficient buffer space. This variable, when combined with ibm3172ifInLANDiscards, reflects the total number of ID frames discarded by the 3172 when no error was detected.	read-only

ibm3172ifDbkCounters Table

Table 37 lists the objects in the IBM 3172 Deblocker counter table. The Deblocker counters identify the transmitted and received outbound octets, blocks, and frames by the LAN adapter. The Deblocker counters also identify the tasks that contained errors and the tasks which were discarded.

Table 37. Implementation of the IBM 3172 Dbk Counter Table

Object	Syntax	Definition	Access
ibm3172ifDbkCounters	ibm3172ifDbkCounters ::= SEQUENCE ibm3172ifDbkRcvOctets COUNTER, ibm3172ifDbkXmitOctets COUNTER, ibm3172ifDbkRcvBlocks COUNTER, ibm3172ifDbkXmitFrames COUNTER, ibm3172ifOutDbkErrors COUNTER, ibm3172ifOutDbkDiscards COUNTER	Objects at the subnetwork layer and below pertaining to a particular Deblocker Task of the 3172. ASN.1 notation: 1.3.6.1.4.1.2.6.1.6	read-only
ibm3172ifDbkRcvOctets {ibm3172ifDbkCounters 1}	Counter	The number of outbound octets that were received by the Deblocker from the channel adapter, including all headers.	read-only

MIB Objects

Table 37. Implementation of the IBM 3172 Dblk Counter Table (continued)

Object	Syntax	Definition	Access
ibm3172ifDblkXmitOctets {ibm3172ifDblkCounters 2}	Counter	The number of outbound octets that were transmitted to the LAN adapter by the Deblocator, including all headers.	read-only
ibm3172ifDblkRcvBlocks {ibm3172ifDblkCounters 3}	Counter	The number of outbound blocks that were received from the channel adapter by the Deblocator Task.	read-only
ibm3172ifDblkXmitFrames {ibm3172ifDblkCounters 4}	Counter	The number of outbound frames that were transmitted to the LAN adapter by the Deblocator Task.	read-only
ibm3172ifOutDblkErrors {ibm3172ifDblkCounters 5}	Counter	The number of outbound frames transmitted by the channel adapter to the Deblocator Task that contained errors preventing them from being deliverable to a higher layer protocol. This variable, when combined with ibm3172ifOutLANErrors, reflects the total number of outbound frames discarded by the 3172 because of errors.	read-only
ibm3172ifOutDblkDiscards {ibm3172ifDblkCounters 6}	Counter	The number of outbound frames transmitted to the Deblocator Task which were discarded by the 3172, even though no errors had been detected to prevent their being deliverable to a higher layer protocol. One possible reason for discarding such a frame could be because of insufficient buffer space. This variable reflects the total number of outbound frames discarded by the 3172 when no error was detected.	read-only

ibm3172ifDeviceTable

Table 38 lists the objects in the IBM 3172 device table. The device table identifies the devices associated with this interface.

Table 38. Implementation of the IBM 3172 Device Table

Object	Syntax	Definition	Access
ibm3172ifDeviceTable	ibm3172ifDeviceTable ::= SEQUENCE	Objects at the interface level pertaining to the associated device.	read-only
	ibm3172ifDeviceNumber INTEGER	ASN.1 notation: 1.3.6.1.4.1.2.6.1.7	
ibm3172ifDeviceNumber {ibm3172ifDeviceTable 1}	Integer	The instance number, which is used to reference the ibm3172SystemTable, for the device associated with this interface.	read-only

Appendix F. IBM 3172 Attribute Index

This appendix lists 3172 attributes and their corresponding MIB variables.

Table 39. MIB Variable Cross Reference Table

3172 Attribute	MIB Variable
01	= ibm3172Descr
02	= ibm3172Contact
03	= ibm3172Location
04	= ibm3172ifNumber
10	= ibm3172ifTrapEnable
11	= ifDescr
12	= ifType
13	= ifPhysAddress
14	= ifOperStatus
20	= ibm3172ifChanCounters
21	= ibm3172ifInChanOctets
22	= ibm3172ifOutChanOctets
23	= ibm3172ifInChanBlocks
24	= ibm3172ifOutChanBlocks
30	= ibm3172ifLANCounters
31	= ibm3172ifInLANOctets
32	= ibm3172ifOutLANOctets
33	= ibm3172ifInLANFrames
34	= ibm3172ifOutLANFrames
35	= ibm3172ifInLANErrors
36	= ibm3172ifOutLANErrors
37	= ibm3172ifInLANDiscards
38	= ibm3172ifOutLANDiscards
40	= ibm3172ifBlkCounters
41	= ibm3172ifBlkRcvOctets
42	= ibm3172ifBlkXmitOctets
43	= ibm3172ifBlkRcvFrames
44	= ibm3172ifBlkXmitBlocks
45	= ibm3172ifInBlkErrors
46	= ibm3172ifInBlkDiscards
50	= ibm3172ifDbkCounters
51	= ibm3172ifDbkRcvOctets
52	= ibm3172ifDbkXmitOctets
53	= ibm3172ifDbkRcvBlocks
54	= ibm3172ifDbkXmitFrames
55	= ibm3172ifOutDbkErrors

Table 39. MIB Variable Cross Reference Table (continued)

3172 Attribute	MIB Variable
56	= ibm3172ifOutDbkDiscards

Appendix G. 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	<p>A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.</p> <p>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.</p>
3	linkUp	<p>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.</p> <p>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.</p>
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	<p>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.</p> <p>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.</p>

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 H. 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	<i>User Datagram Protocol</i>	J.B. Postel
791	<i>Internet Protocol</i>	J.B. Postel
792	<i>Internet Control Message Protocol</i>	J.B. Postel
793	<i>Transmission Control Protocol</i>	J.B. Postel
821	<i>Simple Mail Transfer Protocol</i>	J.B. Postel
822	<i>Standard for the Format of ARPA Internet Text Messages</i>	D. Crocker
823	<i>DARPA Internet Gateway</i>	R.M. Hinden, A. Sheltzer
826	<i>Ethernet Address Resolution Protocol: or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware</i>	D.C. Plummer
854	<i>Telnet Protocol Specification</i>	J.B. Postel, J.K. Reynolds
856	<i>Telnet Binary Transmission</i>	J.B. Postel, J.K. Reynolds
857	<i>Telnet Echo Option</i>	J.B. Postel, J.K. Reynolds
877	<i>Standard for the Transmission of IP Datagrams over Public Data Networks</i>	J.T. Korb
885	<i>Telnet End of Record Option</i>	J.B. Postel
903	<i>Reverse Address Resolution Protocol</i>	R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
904	<i>Exterior Gateway Protocol Formal Specification</i>	D.L. Mills
919	<i>Broadcasting Internet Datagrams</i>	J.C. Mogul
922	<i>Broadcasting Internet Datagrams in the Presence of Subnets</i>	J.C. Mogul
950	<i>Internet Standard Subnetting Procedure</i>	J.C. Mogul, J.B. Postel
952	<i>DoD Internet Host Table Specification</i>	K. Harrenstien, M.K. Stahl, E.J. Feinler
959	<i>File Transfer Protocol</i>	J.B. Postel, J.K. Reynolds
974	<i>Mail Routing and the Domain Name System</i>	C. Partridge
1009	<i>Requirements for Internet Gateways</i>	R.T. Braden, J.B. Postel
1013	<i>X Window System Protocol, Version 11: Alpha Update</i>	R.W. Scheifler
1014	<i>XDR: External Data Representation Standard</i>	Sun Microsystems Incorporated
1027	<i>Using ARP to Implement Transparent Subnet Gateways</i>	S. Carl-Mitchell, J.S. Quarterman
1032	<i>Domain Administrators Guide</i>	M.K. Stahl
1033	<i>Domain Administrators Operations Guide</i>	M. Lottor

RFCs

RFC	Title	Author
1034	<i>Domain Names—Concepts and Facilities</i>	P.V. Mockapetris
1035	<i>Domain Names—Implementation and Specification</i>	P.V. Mockapetris
1042	<i>Standard for the Transmission of IP Datagrams over IEEE 802 Networks</i>	J.B. Postel, J.K. Reynolds
1044	<i>Internet Protocol on Network System's HYPERchannel: Protocol Specification</i>	K. Hardwick, J. Lekashman
1055	<i>Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP</i>	J.L. Romkey
1057	<i>RPC: Remote Procedure Call Protocol Version 2 Specification</i>	Sun Microsystems Incorporated
1058	<i>Routing Information Protocol</i>	C.L. Hedrick
1091	<i>Telnet Terminal-Type Option</i>	J. VanBokkelen
1094	<i>NFS: Network File System Protocol Specification</i>	Sun Microsystems Incorporated
1112	<i>Host Extensions for IP Multicasting</i>	S. Deering
1118	<i>Hitchhikers Guide to the Internet</i>	E. Krol
1122	<i>Requirements for Internet Hosts-Communication Layers</i>	R.T. Braden
1123	<i>Requirements for Internet Hosts-Application and Support</i>	R.T. Braden
1155	<i>Structure and Identification of Management Information for TCP/IP-Based Internets</i>	M.T. Rose, K. McCloghrie
1156	<i>Management Information Base for Network Management of TCP/IP-based Internets</i>	K. McCloghrie, M.T. Rose
1157	<i>Simple Network Management Protocol (SNMP),</i>	J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin
1179	<i>Line Printer Daemon Protocol</i>	The Wollongong Group, L. McLaughlin III
1180	<i>TCP/IP Tutorial,</i>	T. J. Socolofsky, C.J. Kale
1183	<i>New DNS RR Definitions (Updates RFC 1034, RFC 1035)</i>	C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris,
1187	<i>Bulk Table Retrieval with the SNMP</i>	M.T. Rose, K. McCloghrie, J.R. Davin
1188	<i>Proposed Standard for the Transmission of IP Datagrams over FDDI Networks</i>	D. Katz
1198	<i>FYI on the X Window System</i>	R.W. Scheifler
1207	<i>FYI on Questions and Answers: Answers to Commonly Asked Experienced Internet User Questions</i>	G.S. Malkin, A.N. Marine, J.K. Reynolds
1208	<i>Glossary of Networking Terms</i>	O.J. Jacobsen, D.C. Lynch
1213	<i>Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II,</i>	K. McCloghrie, M.T. Rose
1215	<i>Convention for Defining Traps for Use with the SNMP</i>	M.T. Rose
1228	<i>SNMP-DPI Simple Network Management Protocol Distributed Program Interface</i>	G.C. Carpenter, B. Wijnen
1229	<i>Extensions to the Generic-Interface MIB</i>	K. McCloghrie
1230	<i>IEEE 802.4 Token Bus MIB IEEE 802.4 Token Bus MIB</i>	K. McCloghrie, R. Fox
1231	<i>IEEE 802.5 Token Ring MIB IEEE 802.5 Token Ring MIB</i>	K. McCloghrie, R. Fox, E. Decker

RFC	Title	Author
1267	<i>A Border Gateway Protocol 3 (BGP-3)</i>	K. Loughheed, Y. Rekhter
1268	<i>Application of the Border Gateway Protocol in the Internet</i>	Y. Rekhter, P. Gross
1269	<i>Definitions of Managed Objects for the Border Gateway Protocol (Version 3)</i>	S. Willis, J. Burruss
1293	<i>Inverse Address Resolution Protocol</i>	T. Bradley, C. Brown
1270	<i>SNMP Communications Services</i>	F. Kastenholz, ed.
1323	<i>TCP Extensions for High Performance</i>	V. Jacobson, R. Braden, D. Borman
1325	<i>FYI on Questions and Answers: Answers to Commonly Asked New Internet User Questions</i>	G.S. Malkin, A.N. Marine
1350	<i>TFTP Protocol</i>	K.R. Sollins
1351	<i>SNMP Administrative Model</i>	J. Davin, J. Galvin, K. McCloghrie
1352	<i>SNMP Security Protocols</i>	J. Galvin, K. McCloghrie, J. Davin
1353	<i>Definitions of Managed Objects for Administration of SNMP Parties</i>	K. McCloghrie, J. Davin, J. Galvin
1354	<i>IP Forwarding Table MIB</i>	F. Baker
1356	<i>Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode</i>	A. Malis, D. Robinson, R. Ullmann
1374	<i>IP and ARP on HIPPI</i>	J. Renwick, A. Nicholson
1381	<i>SNMP MIB Extension for X.25 LAPB</i>	D. Throop, F. Baker
1382	<i>SNMP MIB Extension for the X.25 Packet Layer</i>	D. Throop
1387	<i>RIP Version 2 Protocol Analysis</i>	G. Malkin
1389	<i>RIP Version 2 MIB Extension</i>	G. Malkin
1390	<i>Transmission of IP and ARP over FDDI Networks</i>	D. Katz
1393	<i>Traceroute Using an IP Option</i>	G. Malkin
1397	<i>Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol</i>	D. Haskin
1398	<i>Definitions of Managed Objects for the Ethernet-like Interface Types</i>	F. Kastenholz
1440	<i>SIFT/UFT:Sender-Initiated/Unsolicited File Transfer</i>	R. Troth
1483	<i>Multiprotocol Encapsulation over ATM Adaptation Layer 5</i>	J. Heinanen
1540	<i>IAB Official Protocol Standards</i>	J.B. Postel
1583	<i>OSPF Version 2</i>	J.Moy
1647	<i>TN3270 Enhancements</i>	B. Kelly
1700	<i>Assigned Numbers</i>	J.K. Reynolds, J.B. Postel
1723	<i>RIP Version 2 — Carrying Additional Information</i>	G. Malkin
1813	<i>NFS Version 3 Protocol Specification</i>	B. Callaghan, B. Pawlowski, P. Stauback, Sun Microsystems Incorporated
2060	<i>IMAP Version 4 Protocol Specification</i>	M. Crispin
2225	<i>Classical IP and ARP over ATM</i>	M. Laubach, J. Halpern
2460	<i>Internet Protocol, Version 6 (IPv6) Specification</i>	S. Deering, R. Hinden

RFCs

RFC	Title	Author
2461	<i>Neighbor Discovery for IP Version 6 (IPv6)</i>	T. Narten, E. Nordmark, W. Simpson
2462	<i>IPv6 Stateless Address Autoconfiguration</i>	S. Thomson, T. Narten
2463	<i>Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification</i>	A. Conta, S. Deering
2710	<i>Multicast Listener Discovery (MLD) for IPv6</i>	S. Deering, W. Fenner, B. Haberman
3484	<i>Default Address Selection for Internet Protocol version 6 (IPv6)</i>	R. Draves
3513	<i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i>	R. Hinden, S. Deering

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 through <http://www.ietf.org/>.

Appendix I. 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
ATM	Asynchronous Transfer Mode
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
CLAW	Common Link Access to Workstation
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
FDDI	Fiber Distributed Data Interface
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
HCH	HYPERchannel device
HIPPI	High Performance Parallel Interface
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force

Abbreviations and Acronyms

IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPL	Initial Program Load
ISA	Industry Standard Adapter
ISDN	Integrated Services Digital Network
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
LPD	Line Printer Daemon
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
TFTP	Trivial File Transfer 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 U.S.A. IBM may not offer the products, services, or features discussed in this document in all 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
Armonk, New York 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, New York 12601-5400
U.S.A.
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information 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 the names and addresses used by an actual business enterprise 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.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

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.

Portions of this product documentation and associated software pertaining to Kerberos software are Copyright (C) 1989 by the Massachusetts Institute of Technology.

Export of this software from the United States of America is assumed to require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.



BSAFE®

Portions of this product documentation and associated software pertaining to the SSL Server use the RSA BSAFE(tm) software Copyright (C) RSA Security Inc.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

AIX	NetView
APL2	OfficeVision
AS/400	OfficeVision/MVS
CICS	OfficeVision/VM
Common User Access	OpenExtensions
CUA	Operating System/2
Database 2	OS/2
DB2	OS/390
DB2 Universal Database	Performance Toolkit for VM
DFSMS/VM	PROFS
DPI	RACF
DYNIX/ptx	RISC System/6000
Enterprise Systems Architecture/390	RS/6000
Enterprise Systems Connection Architecture	SAA
ESCON	SQL/DS
eServer	Systems Application Architecture
GDDM	System/370
HiperSockets	SystemView
IBM	VM/ESA
IBMLink	VSE/ESA
IMS	VTAM
Language Environment	z/Architecture
Micro Channel	z/OS
MVS	z/VM
MVS/ESA	zSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark of The Open Group in the United States or other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary describes the most common terms associated with TCP/IP communication in an internet environment, as used in this book.

For a list of z/VM terms and their definitions, see the *z/VM: Glossary* book.

The glossary is also available through the online HELP Facility. For example, to display the definition of “cms”, enter:

```
help glossary cms
```

You will enter the glossary HELP file and the definition of “cms” will be displayed as the current line. While you are in the glossary HELP file, you can also search for other terms.

If you are unfamiliar with the HELP Facility, you can enter:

```
help
```

to display the main HELP menu, or enter:

```
help cms help
```

for information about the HELP command.

For more information about the HELP Facility, see the *z/VM: CMS User's Guide*.

If you do not find the term you are looking for, see the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

For abbreviations, the definition usually consists only of the words represented by the letters; for complete definitions, see the entries for the words.

Numerics

3172. IBM Interconnect Controller.

3174. IBM Establishment Controller.

3270. Refers to a series of IBM display devices; for example, the IBM 3275, 3276 Controller Display Station, 3277, 3278, and 3279 Display Stations, the 3290 Information Panel, and the 3287 and 3286 printers. A specific device type is used only when a distinction is required between device types. Information about display terminal usage also refers to the IBM 3138, 3148, and 3158 Display Consoles when used in display mode, unless otherwise noted.

37xx Communication Controller. A network interface used to connect a TCP/IP for z/VM or z/OS® network that supports X.25 connections. NCP with X.25 NPSI must be running in the controller, and VTAM must be running on the host.

6611. IBM Network Processor.

8232. IBM LAN Station.

9370. Refers to a series of processors, namely the IBM 9373 Model 20, the IBM 9375 Models 40 and 60, and the IBM 9377 Model 90 and other models.

A

abend. The abnormal termination of a program or task.

abstract syntax. A description of a data structure that is independent of machine-oriented structures and encodings.

Abstract Syntax Notation One (ASN.1). The OSI language for describing abstract syntax.

active gateway. A gateway that is treated like a network interface in that it is expected to exchange routing information, and if it does not do so for a period of time, the route associated with the gateway is deleted.

active open. The state of a connection that is actively seeking a service. Contrast with *passive open*.

adapter. A piece of hardware that connects a computer and an external device. An auxiliary device or unit used to extend the operation of another system.

address. The unique code assigned to each device or workstation connected to a network. 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.

address mask. A bit mask used to select bits from an Internet address for subnet addressing. The mask is 32 bits long and selects the network portion of the Internet address and one or more bits of the local portion. It is sometimes called a subnet mask.

address resolution. A means for mapping network layer addresses onto media-specific addresses. See *ARP*.

Address Resolution Protocol (ARP). A protocol used to dynamically bind an internet address to a hardware

address. ARP is implemented on a single physical network and is limited to networks that support broadcast addressing.

address space. A collection of bytes that are allocated, and in many ways managed, as a single entity by CP. Each byte within an address space is identified by a unique address. An address space represents an extent of storage available to a program. Address spaces allocated by VM range in size from 64KB to 2GB.

Advanced Interactive Executive (AIX). IBM's licensed version of the UNIX operating system.

Advanced Program-to-Program Communications (APPC). The interprogram communication service within SNA LU 6.2 on which the APPC/VM interface is based.

Advanced Research Projects Agency (ARPA). Now called DARPA, its the U.S. Government agency that funded the ARPANET.

Advanced Research Projects Agency Network (ARPANET). A packet switched network developed in the early 1970's that is the forerunner of today's Internet. It was decommissioned in June 1990.

agent. As defined in the SNMP architecture, an agent, or an SNMP server is responsible for performing the network management functions requested by the network management stations.

AIX. Advanced Interactive Executive.

American National Standard Code for Information Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. The default file transfer type for FTP, used to transfer files that contain ASCII text characters.

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States. ANSI is sponsored by the Computer and Business Equipment Manufacturer Association and is responsible for establishing voluntary industry standards.

ANSI. American National Standards Institute.

API. Application Program Interface.

APPC. Advanced Program-to-Program Communications.

application. The use to which an information processing system is put, for example, a payroll application, an airline reservation application, a network application.

application layer. The seventh layer of the OSI (Open Systems Interconnection) model for data communication. It defines protocols for user or application programs.

Application Program Interface (API). The formally defined programming-language interface between an IBM system control program or licensed program and its user. APIs allow programmers to write application programs that use the TCP, UDP, and IP layers of the TCP/IP protocol suite.

argument. A parameter passed between a calling program and a called program.

ARP. Address Resolution Protocol.

ARPA. Advanced Research Projects Agency.

ARPANET. Advanced Research Projects Agency Network.

ASCII. American National Standard Code for Information Interchange. The default file transfer type for FTP, used to transfer files that contain ASCII text characters.

ASN.1. Abstract Syntax Notation One.

ASYNCR. Asynchronous.

asynchronous (ASYNCR). Without regular time relationship; unexpected or unpredictable with respect to the execution of program instruction. See *synchronous*.

asynchronous communication. A method of communication supported by the operating system that allows an exchange of data with remote device, using either a start-stop line or an X.25 line. Asynchronous communications include the file transfer and the interactive terminal facility support.

Athena Widgets. The X Window widget set developed by MIT for Project Athena.

Attachment Unit Interface (AUI). Connector used with thick Ethernet that often includes a drop cable.

AUI. Attachment Unit Interface.

attention key. A function key on terminals that, when pressed, causes an I/O interruption in the processing unit.

authentication server. The service that reads a Kerberos database to verify that a client making a request for access to an end-service is the client named

in the request. The authentication server provides an authenticated client ticket as permission to access the ticket-granting server.

authenticator. Information encrypted by a Kerberos authentication server that a client presents along with a ticket to an end-server as permission to access the service.

authorization. The right granted to a user to communicate with, or to make use of, a computer system or service.

B

backbone. In a local area network multiple-bridge ring configuration, a high-speed link to which rings are connected by means of bridges. A backbone can be configured as a bus or as a ring. In a wide area network, a high-speed link to which nodes or data switching exchanges (DSES) are connected.

background task. A task with which the user is not currently interacting, but continues to run.

baseband. Characteristic of any network technology that uses a single carrier frequency and requires all stations attached to the network to participate in every transmission. See *broadband*.

Basic Encoding Rules (BER). Standard rules for encoding data units described in ASN.1. Sometimes incorrectly grouped under the term ASN.1, which correctly refers only to the abstract description language, not the encoding technique.

Basic Input/Output System (BIOS). A set of routines that permanently resides in read-only memory (ROM) in a PC. The BIOS performs the most basic tasks, such as sending a character to the printer, booting the computer, and reading the keyboard.

batch. An accumulation of data to be processed. A group of records or data processing jobs brought together for processing or transmission. Pertaining to activity involving little or no user action. See *interactive*

Bayonet Neill-Concelman (BNC). A standardized connector used with Thinnet and coaxial cable.

Because It's Time NETWORK (BITNET). A network of hosts that use the Network Job Entry (NJE) protocol to communicate. The network is primarily composed of universities, nonprofit organizations, and research centers. BITNET has recently merged with the Computer and Science Network (CSNET) to form the Corporation for Research and Educational Networking (CSNET). See *CSNET*.

BER. Basic Encoding Rules.

Berkeley Software Distribution (BSD). Term used when describing different versions of the Berkeley UNIX software, as in "4.3BSD UNIX".

BFS. Byte File System.

big-endian. A format for storage or transmission of binary data in which the most significant bit (or byte) comes first. The reverse convention is little-endian.

BIOS. Basic Input/Output System.

BITNET. Because It's Time NETWORK.

Blat. A denial-of-service attack in which the TCP/IP stack is flooded with SYN packets that have spoofed source IP addresses and port numbers that match the destination IP addresses and port numbers. The Blat attack also has the URG flag turned on in the TCP header and has the ability to incrementally spoof the source IP address. Blat is a version of the Land attack.

block. A string of data elements recorded, processed, or transmitted as a unit. The elements can be characters, words, or physical records.

blocking mode. If the execution of the program cannot continue until some event occurs, the operating system suspends the program until that event occurs.

BNC. Bayonet Neill-Concelman.

BOOTPD. Bootstrap Protocol Daemon.

Bootstrap Protocol Daemon (BOOTPD). The BOOTP daemon responds to client requests for boot information using information contained in a BOOTP machine file.

bridge. 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 functional unit that connects two local area networks (LANs) that use the same logical link control (LLC) procedures but may use different medium access control (MAC) procedures.

broadband. Characteristic of any network that multiplexes multiple, independent network carriers onto a single cable. This is usually done using frequency division multiplexing. Broadband technology allows several networks to coexist on one single cable; traffic from one network does not interfere with traffic from another, because the "conversations" happen on different frequencies in the ether, similar to a commercial radio system.

broadcast. The simultaneous transmission of data packets to all nodes on a network or subnetwork.

broadcast address. An address that is common to all nodes on a network.

BSD. Berkeley Software Distribution.

bus topology. A network configuration in which only one path is maintained between stations. Any data transmitted by a station is concurrently available to all other stations on the link.

byte-ordering. The method of sorting bytes under specific machine architectures. Of the two common methods, little endian byte ordering places the least significant byte first. This method is used in Intel** microprocessors. In the second method, big endian byte ordering, the most significant byte is placed first. This method is used in Motorola microprocessors.

Byte File System (BFS). A file system in which a file consists of an ordered sequence of bytes rather than records. BFS files can be organized into hierarchical directories. Byte file systems are enrolled as file spaces in CMS file pools.

C

Carrier Sense Multiple Access with Collision Detection (CSMA/CD). The access method used by local area networking technologies such as Ethernet.

case-sensitive. A condition in which entries for an entry field must conform to a specific lowercase, uppercase, or mixed-case format to be valid.

CCITT. Comite Consultatif International Telegraphique et Telephonique.

CEC.. Central Electronics Complex.

channel. A path in a system that connects a processor and main storage with an I/O device.

channel-attached. pertaining to attachment of devices directly by data channels (I/O channels) to a computer. Pertaining to devices attached to a controlling unit by cables, rather than by telecommunication lines. Synonymous with local, locally attached.

checksum. The sum of a group of data associated with the group and used for checking purposes.

CICS®. Customer Information Control System.

Class A network. An internet network in which the high-order bit of the address is 0. The host number occupies the three, low-order octets.

Class B network. An internet network in which the high-order bit of the address is 1, and the next high-order bit is 0. The host number occupies the two low-order octets.

Class C network. An internet network in which the two high-order bits of the address are 1 and the next high-order bit is 0. The host number occupies the low-order octet.

CLAW. Common Link Access to Workstation.

client. A function that requests services from a server, and makes them available to the user. In z/OS, an address space that is using TCP/IP services.

client-server model. A common way to describe network services and the model user processes (programs) of those services. Examples include the name server and resolver paradigm of the DNS and file server/file client relationships such as NFS and diskless hosts.

client-server relationship. Any device that provides resources or services to other devices on a network is a *server*. Any device that employs the resources provided by a server is a *client*. A machine can run client and server processes at the same time.

CLIST. Command List.

CLPA. Create Link Pack Area.

CMS. Conversational Monitor System.

Comite Consultatif International Telegraphique et Telephonique (CCITT). The International Telegraph and Telephone Consultative Committee. A unit of the International Telecommunications Union (ITU) of the United Nations. CCITT produces technical standards, known as "recommendations," for all internationally controlled aspects of analog and digital communication.

command. The name and any parameters associated with an action that can be performed by a program. The command is entered by the user; the computer performs the action requested by the command name.

Command List (CLIST). A list of commands and statements designed to perform a specific function for the user.

command prompt. A displayed symbol, such as [C:\] that requests input from a user.

Common Link Access to Workstation (CLAW). A continuously executing duplex channel program designed to minimize host interrupts while maximizing channel utilization.

communications adapter. A hardware feature that enables a computer or device to become a part of a data network.

community name. A password used by hosts running Simple Network Management Protocol (SNMP) agents to access remote network management stations.

compile. To translate a program written in a high-level language into a machine language program. The computer actions required to transform a source file into an executable object file.

compiler. A program that translates a source program into an executable program (an object program).

Computer and Science Network (CSNET). A large computer network, mostly in the U.S. but with international connections. CSNET sites include universities, research labs, and some commercial companies. It is now merged with BITNET to form CREN. See *BITNET*.

connection. An association established between functional units for conveying information. The path between two protocol modules that provides reliable stream delivery service. In an internet, a connection extends from a TCP module on one machine to a TCP module on the other.

Control Program (CP). The z/VM operating system that manages the real processor's resources and is responsible for simulating select operating systems, known as virtual machines for individual users. Each virtual machine is the functional equivalent of a *real* machine.

conversational monitor system (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under control of the z/VM control program.

Corporation for Research and Educational Networking (CREN). A large computer network formed from the merging of BITNET and CSNET. See *BITNET* and *CSNET*.

CP. Control Program.

Create Link Pack Area (CLPA). A parameter specified at startup, which says to create the link pack area.

CREN. Corporation for Research and Educational Networking.

CSMA/CD. Carrier Sense Multiple Access with Collision Detection.

CSNET. Computer and Science Network.

Customer Information Control System (CICS). An IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user written application programs. It includes facilities for building, using, and maintaining databases.

D

daemon. A background process usually started at system initialization that runs continuously and performs a function required by other processes. Some daemons are triggered automatically to perform their task; others operate periodically.

DASD. Direct Access Storage Device.

DARPA. Defense Advanced Research Projects Agency.

DATABASE 2 (DB2®). An IBM relational database management system for the z/OS operating system.

database administrator (DBA). An individual or group responsible for the rules by which data is accessed and stored. The DBA is usually responsible for database integrity, security, performance and recovery.

datagram. A basic unit of information that is passed across the internet, it consists of one or more data packets.

data link layer. Layer 2 of the OSI (Open Systems Interconnection) model; it defines protocols governing data packetizing and transmission into and out of each node.

data set. The major unit of data storage and retrieval in z/OS, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. Synonymous with *file* in z/VM.

DB2. DATABASE 2.

DBA. Database administrator.

DBCS. Double Byte Character Set.

DDN. Defense Data Network.

decryption. The unscrambling of data using an algorithm that works under the control of a key. The key allows data to be protected even when the algorithm is unknown. Data is unscrambled after transmission.

default. A value, attribute, or option that is assumed when none is explicitly specified.

Defense Advanced Research Projects Agency (DARPA). The U.S. government agency that funded the ARPANET.

Defense Data Network (DDN). Comprises the MILNET and several other Department of Defense networks.

destination node. The node to which a request or data is sent.

DHCPD. Dynamic Host Configuration Protocol Daemon.

Direct Access Storage Device (DASD). A device in which access to data is independent of where data resides on the device.

directory. A named grouping of files in a file system.

Disk Operating System (DOS). An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

display terminal. An input/output unit by which a user communicates with a data-processing system or sub-system. Usually includes a keyboard and always provides a visual presentation of data; for example, an IBM 3179 display.

Distributed Program Interface (DPI). A programming interface that provides an extension to the function provided by the SNMP agents.

DLL. Dynamic Link Library.

DNS. Domain Name System.

domain. In an internet, a part of the naming hierarchy. Syntactically, a domain name consists of a sequence of names (labels) separated by periods (dots).

Domain Name System (DNS). A system in which a resolver queries name servers for resource records about a host.

domain naming. A hierarchical system for naming network resources.

DoS. Denial-of-Service.

DOS. Disk Operating System.

dotted-decimal notation. The syntactic representation for a 32-bit integer that consists of four 8-bit numbers, written in base 10 and separated by periods (dots). Many internet application programs accept dotted decimal notations in place of destination machine names.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS.

doubleword. A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit.

DPI. Distributed Program Interface.

Dynamic Host Configuration Protocol Daemon (DHCPD). The DHCP daemon (DHCPD server) responds to client requests for boot information using information contained in a DHCP machine file. This information includes the IP address of the client, the IP address of the TFTP daemon, and information about the files to request from the TFTP daemon.

dynamic resource allocation. An allocation technique in which the resources assigned for execution of computer programs are determined by criteria applied at the moment of need.

dynamic link library (DLL). A module containing dynamic link routines that is linked at load or run time.

E

EBCDIC. Extended binary-coded decimal interchange code.

EGP. Exterior Gateway Protocol.

encapsulation. A process used by layered protocols in which a lower-level protocol accepts a message from a higher-level protocol and places it in the data portion of the low-level frame. As an example, in Internet terminology, a packet would contain a header from the physical layer, followed by a header from the network layer (IP), followed by a header from the transport layer (TCP), followed by the application protocol data.

encryption. The scrambling or encoding of data using an algorithm that works under the control of a key. The key allows data to be protected even when the algorithm is unknown. Data is scrambled prior to transmission.

ES/9370 Integrated Adapters. An adapter you can use in TCP/IP for z/VM to connect into Token-Ring networks and Ethernet networks, as well as TCP/IP networks that support X.25 connections.

Ethernet. The name given to a local area packet-switched network technology invented in the early 1970s by Xerox**, Incorporated. Ethernet uses a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) mechanism to send packets.

EXEC. In a VM operating system, a user-written command file that contains CMS commands, other user-written commands, and execution control statements, such as branches.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

extended character. A character other than a 7-bit ASCII character. An extended character can be a 1-bit code point with the 8th bit set (ordinal 128-255) or a 2-bit code point (ordinal 256 and greater).

Exterior Gateway Protocol (EGP). A reachability routing protocol used by gateways in a two-level internet.

eXternal Data Representation (XDR). A standard developed by Sun Microsystems, Incorporated for representing data in machine-independent format.

F

FAT. File Allocation Table.

FDDI. Fiber Distributed Data Interface. Also used to abbreviate Fiber Optic Distributed Data Interface.

Fiber Distributed Data Interface (FDDI). The ANSI standard for high-speed transmission over fiber optic cable.

Fiber Optic Network. A network based on the technology and standards that define data transmission using cables of glass or plastic fibers carrying visible light. Fiber optic network advantages are: higher transmission speeds, greater carrying capacity, and lighter, more compact cable.

file. In z/VM, a named set of records stored or processed as a unit. Synonymous with *data set* in z/OS.

File Allocation Table (FAT). A table used to allocate space on a disk for a file.

File Transfer Access and Management (FTAM). An application service element that enables user application processes to manage and access a file system, which may be distributed.

File Transfer Protocol (FTP). A TCP/IP protocol used for transferring files to and from foreign hosts. FTP also provides the capability to access directories. Password protection is provided as part of the protocol.

foreign host. Any machine on a network that can be interconnected.

foreign network. In an internet, any other network interconnected to the local network by one or more intermediate gateways or routers.

foreign node. See *foreign host*.

Fraggle. A denial-of-service attack in which a UDP Echo Request is sent to a broadcast or multicast address.

frame. The portion of a tape on a line perpendicular to the reference edge, on which binary characters can be written or read simultaneously.

FTAM. File Transfer Access and Management.

FTP. File Transfer Protocol.

fullword. A computer word: 32 bits or 4 bytes.

G

gadget. A windowless graphical object that looks like its equivalent like-named widget but does not support the translations, actions, or pop-up widget children supplied by that widget.

gateway. A functional unit that interconnects a local data network with another network having different protocols. A host that connects a TCP/IP network to a non-TCP/IP network at the application layer. See also 402.

gather and scatter data. Two related operations. During the gather operation, data is taken from multiple buffers and transmitted. In the scatter operation, data is received and stored in multiple buffers.

GC. Graphics Context.

GContext. See *Graphics Context*.

GCS. Group Control System.

GDDM. Graphical Data Display Manager.

GDDMXD. Graphical Data Display Manager interface for X Window System. A graphical interface that formats and displays alphanumeric, data, graphics, and images on workstation display devices that support the X Window System.

GDF. Graphics data file.

Graphical Display Data Manager (GDDM). A group of routines that allows pictures to be defined and displayed procedurally through function routines that correspond to graphic primitives.

Graphics Context (GC). The storage area for graphics output. Also known as *GC* and *GContext*. Used only with graphics that have the same root and depth as the graphics content.

Group Control System (GCS). A component of VM/ESA, consisting of a shared segment that you can Initial Program Load (IPL) and run in a virtual machine. It provides simulated z/OS or OS/390® services and unique supervisor services to help support a native SNA network.

H

handle. A temporary data representation that identifies a file.

halfword. A contiguous sequence of bits or characters that constitutes half a fullword and can be addressed as a unit.

HASP. Houston automatic spooling priority system.

HDLC. High-level Data Link Control.

header file. A file that contains constant declarations, type declarations, and variable declarations and assignments. Header files are supplied with all programming interfaces.

High-level Data Link Control (HDLC). An ISO protocol for X.25 international communication.

High Performance File System (HPFS). An OS/2 file management system that supports high-speed buffer storage, long file names, and extended attributes.

HiperSockets™. A hardware feature that provides high performance internal communications between LPARs within the same CEC.

hop count. The number of gateways or routers through which a packet passes on its way to its destination.

host. A computer connected to a network, which provides an access method to that network. A host provides end-user services and can be a client, a server, or a client and server simultaneously.

Houston automatic spooling priority system (HASP). A computer program that provides supplementary job management, data management, and task management functions such as control of job flow, ordering of tasks, and spooling.

HPFS. High Performance File System.

HYPERchannel Adapter. A network interface used to connect a TCP/IP for z/VM or z/OS host into an existing TCP/IP HYPERchannel network, or to connect TCP/IP hosts together to create a TCP/IP HYPERchannel network.

I

IAB. Internet Activities Board.

ICMP. Internet Control Message Protocol.

IEEE. Institute of Electrical and Electronic Engineers.

IETF. Internet Engineering Task Force.

IGMP. Internet Group Management Protocol (IGMP).

IGP. Interior Gateway Protocol.

include file. A file that contains preprocessor text, which is called by a program, using a standard programming call. Synonymous with *header file*.

IMAP. Internet Message Access Protocol..

IMS™. Information Management System.

Information Management System (IMS). A database/data communication (DB/DC) system that can manage complex databases and networks.

initial program load (IPL). The initialization procedure that causes an operating system to commence operation.

instance. Indicates a label that is used to distinguish among the variations of the *principal name*. An instance allows for the possibility that the same client or service can exist in several forms that require distinct authentication.

Institute of Electrical and Electronic Engineers (IEEE). An electronics industry organization.

Integrated Services Digital Network (ISDN). A digital, end-to-end telecommunication network that supports multiple services including, but not limited to, voice and data.

interactive. Pertaining to a program or a system that alternately accepts input and then responds. An interactive system is conversational; that is, a continuous dialog exists between user and system. See *batch*.

Interior Gateway Protocol (IGP). The protocol used to exchange routing information between collaborating routers in the Internet. RIP is an example of an IGP.

Internet. The largest internet in the world consisting of large national backbone nets (such as MILNET, NSFNET, and CREN) and a myriad of regional and local campus networks all over the world. The Internet uses the Internet protocol suite. To be on the Internet, you must have IP connectivity (be able to TELNET to, or PING, other systems). Networks with only electronic mail connectivity are not actually classified as being on the Internet.

Internet Activities Board (IAB). The technical body that oversees the development of the Internet suite of protocols (commonly referred to as TCP/IP). It has two task forces (the IRTF and the IETF) each charged with investigating a particular area.

Internet address. A 32-bit address assigned to hosts using TCP/IP. An internet address consists of a network number and a local address. Internet addresses are represented in a dotted-decimal notation and are used to route packets through the network.

Internet Engineering Task Force (IETF). One of the task forces of the IAB. The IETF is responsible for solving short-term engineering needs of the Internet.

International Organization for Standardization (ISO). An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

internet or internetwork. A collection of packet switching networks interconnected by gateways, routers, bridges, and hosts to function as a single, coordinated, virtual network.

internet address. The unique 32-bit address identifying each node in an internet. See also 389.

Internet Control Message Protocol (ICMP). The part of the Internet Protocol layer that handles error messages and control messages.

Internet Group Management Protocol (IGMP). IGMP is used by IP hosts to report their host group memberships to multicast routers.

Internet Protocol (IP). The TCP/IP layer between the higher level host-to-host protocol and the local network protocols. IP uses local area network protocols to carry packets, in the form of datagrams, to the next gateway, router, or destination host.

interoperability. The capability of different hardware and software by different vendors to effectively communicate together.

Inter-user communication vehicle (IUCV). A VM facility for passing data between virtual machines and VM components.

intrinsic X-Toolkit. A set management mechanism that provides for constructing and interfacing between composite X Window widgets, their children, and other clients. Also, intrinsic provide the ability to organize a collection of widgets into an application.

IP. Internet Protocol.

IP datagram. The fundamental unit of information passed across the Internet. An IP datagram contains source and destination addresses along with data and a number of fields that define such things as the length of the datagram, the header checksum, and flags to say whether the datagram can be (or has been) fragmented.

IPL. Initial Program Load.

ISDN. Integrated Services Digital Network.

ISO. International Organization for Standardization.

IUCV. Inter-User Communication Vehicle.

J

JCL. Job Control Language.

JES. Job Entry Subsystem.

JIS. Japanese Institute of Standards.

Job Control Language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

Job Entry Subsystem (JES). An IBM licensed program that receives jobs into the system and processes all output data produced by the jobs.

JUNET. The Japanese Academic and Research Network that connects various UNIX operating systems.

K

Kanji. A graphic character set consisting of symbols used in Japanese ideographic alphabets. Each character is represented by 2 bytes.

katakana. A character set of symbols used on one of the two common Japanese phonetic alphabets, which is used primarily to write foreign words phonetically. See also 397.

Kerberos. A system that provides authentication service to users in a network environment.

Kerberos Authentication System. An authentication mechanism used to check authorization at the user level.

Kiss-of-Death (KOD). An IGMP based denial-of-service attack that depletes the stack's large envelopes. See *KOX*.

KOD. Kiss-of-Death.

KOX. An IGMP based denial-of-service attack that depletes the stack's large envelopes and also has source IP address spoofing. *KOX* is a version of the Kiss-of-Death (KOD) attack.

L

LaMail. The client that communicates with the OS/2 Presentation Manager to manage mail on the network.

LAN. Local area network.

Land. A denial-of-service attack in which the TCP/IP stack is flooded with SYN packets that have spoofed source IP addresses and port numbers that match the destination IP addresses and port numbers. See *Blat*.

Line Printer Client (LPR). A client command that allows the local host to submit a file to be printed on a remote print server.

Line Printer Daemon (LPD). The remote printer server that allows other hosts to print on a printer local to your host.

little-endian. A format for storage or transmission of binary data in which the least significant bit (or byte) comes first. The reverse convention is big-endian.

local area network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations.

local host. In an internet, the computer to which a user's terminal is directly connected without using the internet.

local network. The portion of a network that is physically connected to the host without intermediate gateways or routers.

logical character delete symbol. A special editing symbol, usually the at (@) sign, which causes CP to delete it and the immediately preceding character from the input line. If many delete symbols are consecutively entered, the same number of preceding characters are deleted from the input line.

Logical Unit (LU). An entity addressable within an SNA-defined network. LUs are categorized by the types of communication they support.

LPD. Line Printer Daemon.

LPR. Line Printer Client.

LU. Logical Unit.

LU-LU session. In SNA, a session between two logical units (LUs). It provides communication between two end users, or between an end user and an LU services component.

LU type. In SNA, the classification of an LU-LU session in terms of the specific subset of SNA protocols and options supported by the logical units (LUs) for that session.

M

MAC. Media Access Control.

mail gateway. A machine that connects two or more electronic mail systems (often different mail systems on different networks) and transfers messages between them.

Management Information Base (MIB). A standard used to define SNMP objects, such as packet counts and routing tables, that are in a TCP/IP environment.

mapping. The process of relating internet addresses to physical addresses in the network.

mask. A pattern of characters used to control retention or elimination of portions of another pattern of characters. To use a pattern of characters to control retention or elimination of another pattern of characters. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

Maximum Transmission Unit (MTU). The largest possible unit of data that can be sent on a given physical medium.

media access control (MAC). The method used by network adapters to determine which adapter has access to the physical network at a given time.

Message Handling System (MHS). The system of message user agents, message transfer agents, message stores, and access units that together provide OSI electronic mail.

MHS. Message Handling System.

MIB. Management Information Base.

microcode. A code, representing the instructions of an instruction set, which is implemented in a part of storage that is not program-addressable.

MILNET. Military Network.

Military Network (MILNET). Originally part of the ARPANET, MILNET was partitioned in 1984 to make it possible for military installations to have reliable network service, while the ARPANET continued to be used for research. See *DDN*.

minidisk. Logical divisions of a physical direct access storage device.

modem (modulator/demodulator). A device that converts digital data from a computer to an analog signal that can be transmitted on a telecommunication line, and converts the analog signal received to data for the computer.

Motif. see OSF/Motif.

mouse. An input device that is used to move a pointer on the screen and select items.

MPRoute. Multiple Protocol Routing. Implements the OSPF protocol described in RFC 1583, 1058, and 1723.

MTU. Maximum Transmission Unit.

multicast. The simultaneous transmission of data packets to a group of selected nodes on a network or subnetwork.

multiconnection server. A server that is capable of accepting simultaneous, multiple connections.

Multiple Virtual Storage (MVS). Implies the MVS/ESA™, and follow-on OS/390 and z/OS products.

multitasking. A mode of operation that provides for the concurrent performance execution of two or more tasks.

MVS. Multiple Virtual Storage.

N

name server. The server that stores resource records about hosts.

National Science Foundation (NSF). Sponsor of the NSFNET.

National Science Foundation Network (NSFNET). A collection of local, regional, and mid-level networks in the U.S. tied together by a high-speed backbone. NSFNET provides scientists access to a number of supercomputers across the country.

NCP. Network Control Program.

NDB. Network Database.

NDIS. Network Driver Interface Specification.

Netman. This device keyword specifies that this device is a 3172 LAN Channel Station that supports IBM Enterprise-Specific SNMP Management Information Base (MIB) variables for 3172. TCP/IP for VM supports SNMP GET and SNMP GETNEXT operations to request and retrieve 3172 Enterprise-Specific MIB variables. These requests are answered only by those 3172 devices with the NETMAN option in the PROFILE TCPIP file.

NetView®. A system 390-based, IBM-licensed program used to monitor, manage, and diagnose the problems of a network.

network. An arrangement of nodes and connecting branches. Connections are made between data stations. Physical network refers to the hardware that makes up a network. Logical network refers to the abstract organization overlaid on one or more physical networks. An internet is an example of a logical network.

network adapter. A physical device, and its associated software, that enables a processor or controller to be connected to a network.

network administrator. The person responsible for the installation, management, control, and configuration of a network.

Network Control Program (NCP). An IBM-licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

network database (NDB). An IBM-licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. NDB allows interoperability among different database systems, and uses RPC protocol with a client/server type of relationship. NDB is used for data conversion, security, I/O buffer management, and transaction management.

Network Driver Interface Specification (NDIS). An industry-standard specification used by applications as an interface with network adapter device drivers.

network elements. As defined in the SNMP architecture, network elements are gateways, routers, and hosts that contain management agents responsible for performing the network management functions requested by the network management stations.

network file system (NFS). The NFS protocol, which was developed by Sun Microsystems, Incorporated, allows computers in a network to access each other's file systems. Once accessed, the file system appears to reside on the local host.

Network Information Center (NIC). Originally there was only one, located at SRI International and tasked to serve the ARPANET (and later DDN) community. Today, there are many NICs operated by local, regional, and national networks all over the world. Such centers provide user assistance, document service, training, and more.

Network Job Entry (NJE). In object distribution, an entry in the network job table that specifies the system action required for incoming network jobs sent by a particular user or group of users. Each entry is identified by the user ID of the originating user or group.

network layer. Layer 3 of the Open Systems Interconnection (OSI) model; it defines protocols governing data routing.

network management stations. As defined in the SNMP architecture, network management stations, or SNMP clients, execute management applications that monitor and control network elements.

NFS. Network file system.

NIC. Network Information Center.

NJE. Network Job Entry.

node. In a network, a point at which one or more functional units connect channels or data circuits. In a network topology, the point at an end of a branch.

nonblocking mode. If the execution of the program cannot continue until some event occurs, the operating system does not suspend the program until that event occurs. Instead, the operating system returns an error message to the program.

NPSI. X.25 NCP Packet Switching Interface.

NSF. National Science Foundation.

NSFNET. National Science Foundation Network.

O

octet. A byte composed of eight binary elements.

Offload host. Any device that is handling the TCP/IP processing for the z/OS host where TCP/IP for MVS is installed. Currently, the only supported Offload host is the 3172-3.

Offload system. Represents both the z/OS host where TCP/IP for z/OS is installed and the Offload host that is handling the TCP/IP Offload processing.

open system. A system with specified standards and that therefore can be readily connected to other systems that comply with the same standards.

Open Systems Interconnection (OSI). The interconnection of open systems in accordance with specific ISO standards. The use of standardized procedures to enable the interconnection of data processing systems.

Operating System/2 (OS/2). Pertaining to the IBM licensed program that can be used as the operating system for personal computers. The OS/2 licensed program can perform multiple tasks at the same time.

OS/2. Operating System/2.

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 3-D appearance. Performance of the graphic elements is increased with gadgets and windowless widgets.

OSI. Open Systems Interconnection.

OSPF. Open Shortest Path First. An Interior Gateway Protocol that distributes routing information within a single Autonomous System.

out-of-band data. Data that is placed in a secondary channel for transmission. Primary and secondary communication channels are created physically by modulation on a different frequency, or logically by specifying a different logical channel. A primary channel can have a greater capacity than a secondary one.

OV. OfficeVision®.

P

packet. A sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole.

Packet Switching Data Network (PSDN). A network that uses packet switching as a means of transmitting data.

parameter. A variable that is given a constant value for a specified application.

parse. To analyze the operands entered with a command.

passive open. The state of a connection that is prepared to provide a service on demand. Contrast with *active open*.

Partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PC. Personal computer.

PCA. Personal Channel Attach.

PC Network. A low-cost, broadband network that allows attached IBM personal computers, such as IBM 5150 Personal Computers, IBM Computer ATs, IBM PC/XTs, and IBM Portable Personal Computers to communicate and to share resources.

PDS. Partitioned data set.

PDN. Public Data Network.

PDU. Protocol data unit.

peer-to-peer. In network architecture, any functional unit that resides in the same layer as another entity.

Personal Channel Attach (PCA). see Personal System Channel Attach.

Personal Computer (PC). A microcomputer primarily intended for stand-alone use by an individual.

physical layer. Layer 1 of the Open Systems Interconnection (OSI) model; it details protocols governing transmission media and signals.

physical unit (PU). In SNA, the component that manages and monitors the resources, such as attached links and adjacent link stations, associated with a node, as requested by an SSPC via an SSPC-PU session. An SSPC activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links.

PING. The command that sends an ICMP Echo Request packet to a host, gateway, or router with the expectation of receiving a reply.

Ping-o-Death (POD). A denial-of-service attack in which huge, fragmented ICMP packets are sent.

PM. Presentation Manager.

PMANT. In OS/2, the 3270 client terminal emulation program that is invoked by the PMANT command.

polling. On a multipoint connection or a point-to-point connection, the process whereby data stations are invited one at a time to transmit. Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data.

POP. Post Office Protocol.

port. An endpoint for communication between devices, generally referring to a logical connection. A 16-bit number identifying a particular Transmission Control Protocol or User Datagram Protocol resource within a given TCP/IP node.

PORTMAP. Synonymous with *Portmapper*.

Portmapper. A program that maps client programs to the port numbers of server programs. Portmapper is used with Remote Procedure Call (RPC) programs.

Post Office Protocol (POP). A protocol used for exchanging network mail.

presentation layer. Layer 6 of the Open Systems Interconnections (OSI) model; it defines protocols governing data formats and conversions.

Presentation Manager (PM). A component of OS/2 that provides a complete graphics-based user interface, with pull-down windows, action bars, and layered menus.

principal name. Specifies the unique name of a user (client) or service.

PostScript. A standard that defines how text and graphics are presented on printers and display devices.

process. A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. Any operation or combination of operations on data. A function being performed or waiting to be performed. A program in operation; for example, a daemon is a system process that is always running on the system.

Professional Office Systems (PROFS®). IBM's proprietary, integrated office management system used for sending, receiving, and filing electronic mail, and a variety of other office tasks. PROFS has been replaced by OfficeVision. See *OfficeVision*.

PROFS. Professional Office Systems.

protocol. A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. Protocols can determine low-level details of machine-to-machine interfaces, such as the order in which bits from a byte are sent; they can also determine high-level exchanges between application programs, such as file transfer.

Protocol data unit (PDU). A set of commands used by the SNMP agent to request management station data.

protocol suite. A set of protocols that cooperate to handle the transmission tasks for a data communication system.

PSDN. Packet Switching Data Network.

PU. Physical unit.

Public Data Network (PDN). A network established and operated by a telecommunication administration or by a Recognized Private Operating Agency (RPOA) for the specific purpose of providing circuit-switched, packet-switched, and leased-circuit services to the public.

Q

QDIO. Queued Direct I/O.

queue. A line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted. To arrange in, or form, a queue.

R

R4P3D. A denial-of-service attack in which TCP packets are sent to the stack with no header flags set. R4P3D is an augmented version of the Stream attack.

RACF. Resource access control facility.

RARP. Reverse Address Resolution Protocol.

read-only access. An access mode associated with a virtual disk directory that lets a user read, but not write or update, any file on the disk directory.

read/write access. An access mode associated with a virtual disk directory that lets a user read and write any file on the disk directory (if write authorized).

realm. One of the three parts of a Kerberos name. The realm specifies the network address of the principal name or instance. This address must be expressed as a fully qualified domain name, not as a "dot numeric" internet address.

recursion. A process involving numerous steps, in which the output of each step is used for the successive step.

reduced instruction-set computer (RISC). A computer that uses a small, simplified set of frequently used instructions for rapid execution.

reentrant. The attribute of a program or routine that allows the same copy of a program or routine to be used concurrently by two or more tasks.

Remote Execution Protocol (REXEC). A protocol that allows the execution of a command or program on a foreign host. The local host receives the results of the command execution. This protocol uses the REXEC command.

remote host. A machine on a network that requires a physical link to interconnect with the network.

remote logon. The process by which a terminal user establishes a terminal session with a remote host.

Remote Procedure Call (RPC). A facility that a client uses to request the execution of a procedure call from a server. This facility includes a library of procedures and an eXternal data representation.

Remote Spooling Communications Subsystem (RSCS). An IBM-licensed program that transfers spool files, commands, and messages between VM users, remote stations, and remote and local batch systems, through HASP-compatible telecommunication facilities.

Request For Comments (RFC). A series of documents that covers a broad range of topics affecting internetwork communication. Some RFCs are established as internet standards.

resolver. A program or subroutine that obtains information from a name server or local table for use by the calling program.

resource access control facility (RACF). An IBM-licensed program that provides for access control by identifying and by verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource records. Individual records of data used by the Domain Name System. Examples of resource records include the following: a host's Internet Protocol addresses, preferred mail addresses, and aliases.

response unit (RU). In SNA, a message unit that acknowledges a request unit. It may contain prefix information received in a request unit. If positive, the response unit may contain additional information such as session parameters in response to BIND SESSION. If negative, it contains sense data defining the exception condition.

Restructured Extended Executor (REXX) language. A general purpose programming language, particularly suitable for EXEC procedures, XEDIT macros, or programs for personal computing. Procedures, XEDIT

macros, and programs written in this language can be interpreted by the Procedures Language VM/REXX interpreter.

return code. A code used to influence the execution of succeeding instructions. A value returned to a program to indicate the results of an operation requested by that program.

Reverse Address Resolution Protocol (RARP). A protocol that maintains a database of mappings between physical hardware addresses and IP addresses.

REXEC. Remote Execution Protocol.

REXX. Restructured Extended Executor language.

RFC. Request For Comments.

RIP. Routing Information Protocol.

RISC. Reduced instruction-set computer.

ROUTED.. Routing Daemon.

router. 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. In TCP/IP, routers operate at the Internetwork layer. See also 395.

Routing Information Protocol (RIP). The protocol that maintains routing table entries for gateways, routers, and hosts.

routing table. A list of network numbers and the information needed to route packets to each.

RPC. Remote Procedure Call.

RSCS. Remote Spooling Communications Subsystem.

RU. Response unit.

S

SAA. Systems Application Architecture.

SBCS. Single Byte Character Set.

Sendmail. The OS/2 mail server that uses Simple Mail Transfer Protocol to route mail from one host to another host on the network.

serial line. A network media that is a de facto standard, not an international standard, commonly used for point-to-point TCP/IP connections. Generally, a serial line consists of an RS-232 connection into a modem and over a telephone line.

semantics. The relationships of characters or groups of characters to their meanings, independent of the manner of their interpretation and use. The relationships between symbols and their meanings.

server. A function that provides services for users. A machine can run client and server processes at the same time.

SFS. Shared File System.

Shared File System (SFS). A part of CMS that lets users organize their files into groups known as *directories* and selectively share those files and directories with other users.

Simple Mail Transfer Protocol (SMTP). A TCP/IP application protocol used to transfer mail between users on different systems. SMTP specifies how mail systems interact and the format of control messages they use to transfer mail.

Simple Network Management Protocol (SNMP). A protocol that allows network management by elements, such as gateways, routers, and hosts. This protocol provides a means of communication between network elements regarding network resources.

simultaneous peripheral operations online (SPOOL). (Noun) An area of auxiliary storage defined to temporarily hold data during its transfer between peripheral equipment and the processor. (Verb) To use auxiliary storage as a buffer storage to reduce processing delays when transferring data between peripheral equipment and the processing storage of a computer.

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with double-byte character set.

SMI. Structure for Management Information.

SMTP. Simple Mail Transfer Protocol.

Smurf. A denial-of-service attack in which an ICMP Echo Request is sent to a broadcast or multicast address. There are three variants of the Smurf attack. See *Smurf-IC*, *Smurf-OB*, and *Smurf-RP*.

Smurf-IC. A denial-of-service attack in which an ICMP Echo Request is sent to a broadcast or multicast address. "IC" denotes that incoming packets are using the TCP/IP stack to launch an attack. See *Smurf-OB* and *Smurf-RP*.

Smurf-OB. A denial-of-service attack in which an ICMP Echo Request is sent to a broadcast or multicast address. "OB" denotes that an outbound ICMP Echo Request matched the description of a Smurf attack. See *Smurf-IC* and *Smurf-RP*.

Smurf-RP. A denial-of-service attack in which an ICMP Echo Request is sent to a broadcast or multicast address. "RP" denotes that the ICMP Echo Reply packets being received by the stack do not match any Echo Requests that were sent. See *Smurf-IC* and *Smurf-OB*.

SNA. Systems Network Architecture.

SNALINK. SNA Network Link.

SNA Network Link. An SNA network link function of TCP/IP for z/VM and OS/390 hosts running TCP/IP to communicate through an existing SNA backbone.

SNMP. Simple Network Management Protocol.

SOA. Start of authority record.

socket. An endpoint for communication between processes or applications. A pair consisting of TCP port and IP address, or UDP port and IP address.

socket address. An address that results when the port identification number is combined with an internet address.

socket interface. An application interface that allows users to write their own applications to supplement those supplied by TCP/IP.

spoofing. An act of forging and inserting data that is incorrect or not valid. It is most commonly used in reference to IP source spoofing, where the source address in an IP packet header is replaced with a false one, effectively masking the source of the packet (making it difficult to trace back to the originator).

SPOOL. Simultaneous peripheral operations online.

spooling. The processing of files created by or intended for virtual readers, punches, and printers. The spool files can be sent from one virtual device to another, from one virtual machine to another, and to read devices.

SQL. Structured Query Language.

SQL/DS™. Structured Query Language/Data System.

SSL. Secure Sockets Layer. Provides the secure (encrypted) communication between a remote client and a TCP/IP server.

start of authority record (SOA). In the Domain Name System, the resource record that defines a zone.

stream. A continuous sequence of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

Stream. A denial-of-service attack in which TCP packets are sent to the stack with no header flags set. See *R4P3D*.

Structured Query Language (SQL). Fourth generation English-like programming language used to perform queries on relational databases.

Structured Query Language/Data System (SQL/DS). An IBM relational database management system for the VM and VSE operating systems.

Structure for Management Information (SMI). The rules used to define the objects that can be accessed through a network management protocol. See also 398.

subdirectory. A directory contained within another directory in a file system hierarchy.

subnet. A networking scheme that divides a single logical network into smaller physical networks to simplify routing.

subnet address. The portion of the host address that identifies a subnetwork.

subnet mask. A mask used in the IP protocol layer to separate the subnet address from the host portion of the address.

subnetwork. Synonymous with *subnet*.

subsystem. A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system.

SYNC. Synchronous.

synchronous (SYNC). Pertaining to two or more processes that depend on the occurrences of a specific event such as common timing signal. Occurring with a regular or predictable time relationship. See *asynchronous*.

SynFlood. A denial-of-service attack in which the initiator floods the TCP/IP stack with SYN packets that have spoofed source IP addresses, resulting in the server never receiving the final ACKs needed to complete the three-way handshake in the connection process.

Systems Application Architecture (SAA). A formal set of rules that enables applications to be run without modification in different computer environments.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

T

TALK. An interactive messaging system that sends messages between the local host and a foreign host.

TCP. Transmission Control Protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol

Telnet. The Terminal Emulation Protocol, a TCP/IP application protocol for remote connection service. Telnet allows a user at one site to gain access to a foreign host as if the user's terminal were connected directly to that foreign host.

terminal emulator. A program that imitates the function of a particular kind of terminal.

Terminate and Stay Resident (TSR) program. A TSR is a program that installs part of itself as an extension of DOS when it is executed.

TFTPD. Trivial File Transfer Protocol Daemon.

ticket. Encrypted information obtained from a Kerberos authentication server or a ticket-granting server. A ticket authenticates a user and, in conjunction with an authenticator, serves as permission to access a service when presented by the authenticated user.

ticket-granting server. Grants Kerberos tickets to authenticated users as permission to access an end-service.

Time Sharing Option (TSO). An operating system option; for z/OS, the option provides interactive time sharing from remote terminals

time stamp. To apply the current system time. The value on an object that is an indication of the system time at some critical point in the history of the object. In query, the identification of the day and time when a query report was created that query automatically provides on each report.

TN3270. An informally defined protocol for transmitting 3270 data streams over Telnet.

token. In a local network, the symbol of authority passed among data stations to indicate the station temporarily in control of the transmission medium.

token-bus. See *bus topology*.

token ring. As defined in IEEE 802.5, a communication method that uses a token to control access to the LAN. The difference between a token bus and a token ring is that a token-ring LAN does not use a master controller to control the token. Instead, each computer knows the address of the computer that should receive the token next. When a computer with the token has nothing to transmit, it passes the token to the next computer in line.

token-ring network. A ring network that allows unidirectional data transmission between data stations by a token-passing procedure over one transmission medium, so that the transmitted data returns to the transmitting station.

Transmission Control Protocol (TCP). The TCP/IP layer that provides reliable, process-to-process data stream delivery between nodes in interconnected computer networks. TCP assumes that IP (Internet Protocol) is the underlying protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP). A suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network.

transport layer. Layer 4 of the Open Systems Interconnection (OSI) model; it defines protocols governing message structure and some error checking.

TRAP. An unsolicited message that is sent by an SNMP agent to an SNMP network management station.

Trivial File Transfer Protocol Daemon (TFTP). The TFTP daemon (TFTPD server) transfers files between the Byte File System (BFS) and TFTP clients. TFTPD supports access to files maintained in a BFS directory structure that is mounted.

TSO. Time Sharing Option.

TSR. Terminate and stay resident. TSR usually refers to a terminate-and-stay-resident program.

U

UDP. User Datagram Protocol.

user. A function that uses the services provided by a server. A host can be a user and a server at the same time. See *client*.

User Datagram Protocol (UDP). A datagram level protocol built directly on the IP layer. UDP is used for application-to-application programs between TCP/IP hosts.

user exit. A point in an IBM-supplied program at which a user routine may be given control.

user profile. A description of a user, including user ID, user name, defaults, password, access authorization, and attributes.

V

virtual address. The address of a location in virtual storage. A virtual address must be translated into a real address to process the data in processor storage.

Virtual Machine (VM). Licensed software whose full name is Virtual Machine/Enterprise Systems Architecture (VM/ESA). It is a software operating system that manages the resources of a real processor to provide virtual machines to end users. It includes time-sharing system control program (CP), the

conversational monitor system (CMS), the group control system (GCS), and the dump viewing facility (DVF).

Virtual Machine Communication Facility (VMCF). A connectionless mechanism for communication between address spaces.

VM. Virtual machine.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, not by the actual number of main storage locations.

Virtual Telecommunications Access Method (VTAM). An IBM-licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

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.

W

WAN. Wide area network.

well-known port. A port number that has been preassigned for specific use by a specific protocol or application. Clients and servers using the same protocol communicate over the same well-known port.

wide area network (WAN). A network that provides communication services to a geographic area larger than that served by a local area network.

widget. The basic data type of the X Window System Toolkit. Every widget belongs to a widget class that contains the allowed operations for that corresponding class.

window. An area of the screen with visible boundaries through which a panel or portion of a panel is displayed.

working directory. The directory in which an application program is found. The working directory becomes the current directory when the application is started.

X

X Client. An application program which uses the X protocol to communicate windowing and graphics requests to an X Server.

XDR. eXternal Data Representation.

XEDIT. The CMS facility, containing the XEDIT command and XEDIT subcommands and macros, that lets a user create, change, and manipulate CMS files.

X Server. A program which interprets the X protocol and controls one or more screens, a pointing device, a keyboard, and various resources associated with the X Window System, such as Graphics Contexts, Pixmap, and color tables.

X Window System. The X Window System is a protocol designed to support network transparent windowing and graphics. TCP/IP for z/VM and OS/390 provides client support for the X Window System application program interface.

X Window System API. An application program interface designed as a distributed, network-transparent, device-independent, windowing and graphics system.

X Window System Toolkit. Functions for developing application environments.

X.25. A CCITT communication protocol that defines the interface between data terminal equipment and packet switching networks.

X.25 NCP Packet Switching Interface (X.25 NPSI).

An IBM-licensed program that allows users to communicate over packet switched data networks that have interfaces complying with Recommendation X.25 (Geneva** 1980) of the CCITT. It allows SNA programs to communicate with SNA equipment or with non-SNA equipment over such networks.

Z

ZAP. To modify or dump an individual text file/data set using the ZAP command or the ZAPTEXT EXEC.

zone. In the Domain Name System, a zone is a logical grouping of domain names that is assigned to a particular organization. Once an organization controls its own zone, it can change the data in the zone, add new tree sections connected to the zone, delete existing nodes, or delegate new subzones under its zone.

Bibliography

This bibliography lists the books in the z/VM product library. For abstracts of these books and information about current editions and available media, see *z/VM: General Information*.

Where to Get z/VM Books

z/VM books are available from the following sources:

- IBM Publications Center at
www.ibm.com/shop/publications/order/
- z/VM Internet Library at
www.ibm.com/eserver/zseries/zvm/library/
- *IBM Online Library: z/VM Collection*, SK2T-2067
- *IBM Online Library: z/VM Collection on DVD*, SK5T-7054

z/VM Base Library

The following books describe the facilities included in the z/VM base product.

Overview

- z/VM: General Information*, GC24-6095
- z/VM: Glossary*, GC24-6097
- z/VM: License Information*, GC24-6102

Installation, Migration, and Service

- z/VM: Guide for Automated Installation and Service*, GC24-6099
- z/VM: Migration Guide*, GC24-6103
- z/VM: Service Guide*, GC24-6117
- z/VM: VMSES/E Introduction and Reference*, GC24-6130

Planning and Administration

- z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6074
- z/VM: CMS Planning and Administration*, SC24-6078
- z/VM: Connectivity*, SC24-6080
- z/VM: CP Planning and Administration*, SC24-6083
- z/VM: Getting Started with Linux on System z9 and zSeries*, SC24-6096
- z/VM: Group Control System*, SC24-6098

z/VM: I/O Configuration, SC24-6100

z/VM: Running Guest Operating Systems, SC24-6115

z/VM: Saved Segments Planning and Administration, SC24-6116

z/VM: TCP/IP Planning and Customization, SC24-6125

eServer zSeries 900: Planning for the Open Systems Adapter-2 Feature, GA22-7477

System z9 and eServer zSeries: Open Systems Adapter-Express Customer's Guide and Reference, SA22-7935

System z9 and eServer zSeries: Open Systems Adapter-Express Integrated Console Controller User's Guide, SA22-7990

z/OS and z/VM: Hardware Configuration Manager User's Guide, SC33-7989

Customization and Tuning

- z/VM: CP Exit Customization*, SC24-6082
- z/VM: Performance*, SC24-6109

Operation

- z/VM: System Operation*, SC24-6121
- z/VM: Virtual Machine Operation*, SC24-6128

Application Programming

- z/VM: CMS Application Development Guide*, SC24-6069
- z/VM: CMS Application Development Guide for Assembler*, SC24-6070
- z/VM: CMS Application Multitasking*, SC24-6071
- z/VM: CMS Callable Services Reference*, SC24-6072
- z/VM: CMS Macros and Functions Reference*, SC24-6075
- z/VM: CP Programming Services*, SC24-6084
- z/VM: CPI Communications User's Guide*, SC24-6085
- z/VM: Enterprise Systems Architecture/Extended Configuration Principles of Operation*, SC24-6094
- z/VM: Language Environment User's Guide*, SC24-6101
- z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6104

z/VM: OpenExtensions Callable Services Reference, SC24-6105

z/VM: OpenExtensions Commands Reference, SC24-6106

z/VM: OpenExtensions POSIX Conformance Document, GC24-6107

z/VM: OpenExtensions User's Guide, SC24-6108

z/VM: Program Management Binder for CMS, SC24-6110

z/VM: Reusable Server Kernel Programmer's Guide and Reference, SC24-6112

z/VM: REXX/VM Reference, SC24-6113

z/VM: REXX/VM User's Guide, SC24-6114

z/VM: Systems Management Application Programming, SC24-6122

z/VM: TCP/IP Programmer's Reference, SC24-6126

Common Programming Interface Communications Reference, SC26-4399

Common Programming Interface Resource Recovery Reference, SC31-6821

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 ILC Applications, SA22-7563

z/OS MVS Program Management: Advanced Facilities, SA22-7644

z/OS MVS Program Management: User's Guide and Reference, SA22-7643

End Use

z/VM: CMS Commands and Utilities Reference, SC24-6073

z/VM: CMS Pipelines Reference, SC24-6076

z/VM: CMS Pipelines User's Guide, SC24-6077

z/VM: CMS Primer, SC24-6137

z/VM: CMS User's Guide, SC24-6079

z/VM: CP Commands and Utilities Reference, SC24-6081

z/VM: Quick Reference, SC24-6111

z/VM: TCP/IP User's Guide, SC24-6127

z/VM: XEDIT Commands and Macros Reference, SC24-6131

z/VM: XEDIT User's Guide, SC24-6132

CMS/TSO Pipelines Author's Edition, SL26-0018

System Diagnosis

z/VM: CMS and REXX/VM Messages and Codes, GC24-6118

z/VM: CP Messages and Codes, GC24-6119

z/VM: Diagnosis Guide, GC24-6092

z/VM: Dump Viewing Facility, GC24-6093

z/VM: Other Components Messages and Codes, GC24-6120

z/VM: TCP/IP Diagnosis Guide, GC24-6123

z/VM: TCP/IP Messages and Codes, GC24-6124

z/VM: VM Dump Tool, GC24-6129

z/OS and z/VM: Hardware Configuration Definition Messages, SC33-7986

Books for z/VM Optional Features

The following books describe the optional features of z/VM.

Data Facility Storage Management Subsystem for VM

z/VM: DFSMS/VM Customization, SC24-6086

z/VM: DFSMS/VM Diagnosis Guide, GC24-6087

z/VM: DFSMS/VM Messages and Codes, GC24-6088

z/VM: DFSMS/VM Planning Guide, SC24-6089

z/VM: DFSMS/VM Removable Media Services, SC24-6090

z/VM: DFSMS/VM Storage Administration, SC24-6091

Directory Maintenance Facility

z/VM: Directory Maintenance Facility Commands Reference, SC24-6133

z/VM: Directory Maintenance Facility Messages, GC24-6134

z/VM: Directory Maintenance Facility Tailoring and Administration Guide, SC24-6135

Performance Toolkit for VM™

z/VM: Performance Toolkit, SC24-6136

Resource Access Control Facility

*External Security Interface (RACROUTE)
Macro Reference for MVS and VM*,
GC28-1366

*Resource Access Control Facility: Auditor's
Guide*, SC28-1342

*Resource Access Control Facility: Command
Language Reference*, SC28-0733

*Resource Access Control Facility: Diagnosis
Guide*, GY28-1016

*Resource Access Control Facility: General
Information*, GC28-0722

*Resource Access Control Facility: General
User's Guide*, SC28-1341

*Resource Access Control Facility: Macros and
Interfaces*, SC28-1345

*Resource Access Control Facility: Messages
and Codes*, SC38-1014

*Resource Access Control Facility: Migration
and Planning*, GC23-3054

*Resource Access Control Facility: Security
Administrator's Guide*, SC28-1340

*Resource Access Control Facility: System
Programmer's Guide*, SC28-1343

- *"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. Warriar.
- "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.

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM V5.2 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.

Index

Numerics

3172 attributes 369
3172 enterprise-specific MIB variables 362

A

abbreviations and acronyms 377
ACCT (FTP subcommand) 51
address fields
 class A 14
 class B 14
 class C 15
 class D 15
Address Resolution Protocol (ARP) 7
AIX files 323
ALL (NETSTAT parameter) 123, 135
ALLCONN (NETSTAT parameter) 124, 135
AO (TELNET subcommand) 117
APL2 Character Set 329
APPEND (FTP subcommand) 52
application layer 4
applications, functions and protocols
 Bootstrap Protocol (BOOTP) 12
 Domain Name System (DNS) 10, 277
 Dynamic Host Configuration Protocol (DHCP) 12
 File Transfer Protocol (FTP) 9, 27
 GDDMXD 11
 Kerberos Authentication System 11, 163, 169
 Network Database System (NDB) 12, 265
 Network File System (NFS) 12, 169
 Remote Execution Protocol (REXEC) 12, 201
 Remote Printing (LPD and LPR) 11, 207
 Remote Procedure Call (RPC) 12
 Routed 11
 Simple Mail Transfer Protocol (SMTP) 10
 Simple Network Management Protocol (SNMP) 10, 247
 Socket interfaces 13
 Telnet Protocol 9, 115
 Trivial File Transfer Protocol (TFTP) 9, 97
 X Toolkit 11
 X Window System 11
ARP (NETSTAT parameter) 136
AS 400 files 326
ASCII (FTP subcommand) 52
ASCII mode
 with the TFTP GET subcommand 98
 with the TFTP MODE subcommand 100
ASCII-to-EBCDIC table 316
Athena widget 11
attacks
 Fraggle 129
 Ping-o-Death 129
 Smurf 129
attacks, denial-of-service (DOS)
 example 145
authentication of network users 163

AYT (TELNET subcommand) 117

B

BINARY (FTP subcommand) 53
bit mask 16
BLOCK (NETSTAT parameter) 124
Bootstrap Protocol (BOOTP) 12
bridge 2
broadcast address format 15

C

CD (FTP subcommand) 54
changing the FTP file transfer type
 to ASCII 52
 to Image 53
 to Kanji 64, 83
character sets 311
client 2
CLIENTS (NETSTAT parameter) 125, 137
CLOSE (FTP subcommand) 57
CMS (FTP subcommand) 57
CMS command interface 306
CMSRESOL 305
code pages 311
computer networks
 LAN 1
 WAN 1
CONN (NETSTAT parameter) 126, 141
connecting to a database
 explicit connection 269
 implicit connection 269
Controlling File Translation (FTP) 45
CONVXLAT command 321
CP (NETSTAT parameter) 126, 141
CP MSG command 190, 191, 192
customizing
 DBCS translation tables 317
 SBCS translation tables 316

D

data set names 323
database query and update 308
datagram 2
datagram socket 248
DBCS facilities 335
DEBUG (FTP subcommand) 58
DELARP (NETSTAT parameter) 126
DELETE (FTP subcommand) 58
deleting CMS record-length fields 198
DELIMIT (FTP subcommand) 59
DELNeighbor (NETSTAT parameter) 126
denial-of-service (DOS) attacks
 example 145
 Fraggle 129

- denial-of-service (DOS) attacks *(continued)*
 - Ping-o-Death 129
 - Smurf 129
- DEVLINKS (NETSTAT parameter) 127, 142
- DiG command 294
- DiG internal state information 294
- DiG options 298
- DiG program 294
- DIR (FTP subcommand) 59
- direct routing 13
- domain name servers 278
- Domain Name System (DNS)
 - CMS command interface to the resolver 306
 - CMSRESOL, resolver and name server 305
 - NSLOOKUP examples 291
 - Overview of Domain Name System 10, 277
 - query types 307
 - Querying Name Servers-DiG 294
 - Querying Name Servers-NSLOOKUP 283
 - resolvers 279
 - resource records 280
 - Using the Domain Name System 277
- domain names 277
- DOS names 325
- dotted-decimal notation 121
- DROP (NETSTAT parameter) 129, 146
- Dynamic Host Configuration Protocol (DHCP) 12

E

- EBCDIC (FTP subcommand) 61
- EBCDIC-to-ASCII table 317
- electronic mail
 - delivery 106
 - gateway 105
 - notes
 - non-delivery note 107
 - unknown recipient note 107
 - notes without PROFS interface 110
 - PROFS interface 108
 - SMTPQUEUE command 106
 - TCP/IP recipients 108
- EUCKANJI (FTP subcommand) 61
- Extended Mail, PROFS 108
- eXternal Data Representation (XDR) 267

F

- file name translation (NFS)
 - special names 196
- file names 323
- file naming format, FTP 41
- File Transfer Protocol (FTP) 9, 27, 96
- file transfer type
 - ASCII 45, 52, 87
 - EBCDIC 45, 61, 87, 88
 - Image 45, 53, 87
 - Kanji 45, 87, 88
- FILTERS (NETSTAT parameter) 129
- foreign host 2
- Fraggle attack 129

- FTP
 - file transfer methods 45
 - naming files 41
- FTP command format 28
- FTP DATA File 31
 - Configuration Statements 32
 - CCONNTIME Statement 32
 - DATACTTIME Statement 33
 - DCONNTIME Statement 34
 - INACTTIME Statement 35
 - LOADDBCSTABLE Statement 36
 - MYOPENTIME Statement 37
 - Statement Syntax 31
- FTP data transfer methods 45
- FTP EXEC interface 92
- FTP EXIT return codes 92
- FTP Kanji support 335
- FTP parameters
 - EXIT 28
 - foreign host 28
 - port number 28
 - TIMEOUT 29
 - TRACE 29
 - TRANSLATE 29
- FTP reply codes 94
- FTP servers, RACF support 96
- FTP subcommands
 - ACCT 51
 - APPEND 52
 - ASCII 52
 - BINARY 53
 - CD 54, 56
 - CDUP 57
 - CLOSE 57
 - CMS 57
 - DEBUG 58
 - DELETE 58
 - DELIMIT 59
 - DIR 59
 - EBCDIC 61
 - EUCKANJI 61
 - GET 61
 - HELP 63
 - IBMKANJI 63
 - JIS78KJ 64
 - JIS83KJ 64
 - LCD 65
 - LOCSITE 66
 - LOCSTAT 66
 - LPWD 67
 - LS 67
 - MDELETE 69
 - MGET 69
 - MKDIR 70
 - MODE 71
 - MPUT 71
 - NETRC 72
 - NOOP 73
 - OPEN 73
 - PASS 74
 - PASSIVE 75

FTP subcommands (*continued*)

- PUT 75
- PWD 76
- QUIT 77
- QUOTE 77
- RENAME 78
- SENDPORT 79
- SENDSITE 80
- SITE 80
- SIZE 83
- SJISKANJI 83
- STATUS 83
- STRUCT 84
- SUNIQUE 84
- SYSTEM 86
- TRACE 29
- TRANSLATE 29
- TYPE 87
- USER 88

G

- GATE (NETSTAT parameter) 129, 146
- gateway 2
- gddmx*ANFont command 239
- gddmx*CMap: command 236
- gddmx*Compr: command 238
- gddmx*Enter: command 239
- gddmx*GColonn: command 234
- gddmx*Geometry: command 233
- gddmx*GMCPnn: command 235
- gddmx*HostRast: command 237
- gddmx*NewLine: command 240
- gddmx*PDTrace: command 238
- gddmx*XCiConn: command 237
- gddmx*XSync: command 236
- gddmx*ZWL: command 235
- GDDMXD 11, 229, 247, 329
- GDDMXD command 230
- GDDMXD/VM
 - CMS GLOBALV command format 231
 - command format 230
 - demonstration programs
 - GXDEMO1 244
 - GXDEMO2 244
 - GXDEMO3 244
 - GXDEMO4 244
 - GXDEMO5 244
 - GXDEMO6 244
 - GXDEMO7 245
 - GDXAPLCS MAP 242
 - keyboard functions 240, 241
 - limitations 229
 - overview 229
 - problem determination 242, 243
 - trace 243
 - user-specified options
 - CMap 236
 - GColonn 234
 - Geometry 233
 - GMCPnn 235

GDDMXD/VM (*continued*)

user-specified options (*continued*)

- HostRast 237
- XCiConn 237
- XSync 236
- ZWL 235
- VM/XA considerations 230
- X DEFAULTS 232
- GET (FTP subcommand) 61
- GET (TFTP subcommand) 98
- GLOBALV command 231
- glossary information 389

H

- HELP (FTP subcommand) 63
- HELP (NETSTAT subcommand) 130, 147
- HELP (TELNET subcommand) 118
- HELP (TFTP subcommand) 99
- HOME (NETSTAT parameter) 130, 148
- host
 - foreign 2
 - local 2

I

- IBMKANJI (FTP subcommand) 63
- IDENTIFY (NETSTAT subcommand) 130, 150
- IMAP
 - using 110
- indirect routing 13
- Integrated Services Digital Network (ISDN) 1
- internal error codes 95
- International Telegraph and Telephone Consultative Committee (CCITT) 6
- internet address 2
- internet addressing
 - broadcast address format 15
 - internet address 2
 - local address 14
 - network address format 14
 - network number 14
 - subnetwork address format 16
 - subnetwork number 14
- 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

- Internet Environment (*continued*)
 - router 2
 - server 2
 - user 2
- Internet Message Access Protocol (IMAP) 10
- Internet Protocol (IP) 7
- internetwork layer 4
- internetwork protocols 6
 - Address Resolution Protocol (ARP) 7
 - Internet Control Message Protocol (ICMP) 7
 - Internet Protocol (IP) 7
- INTERVAL (NETSTAT parameter) 130, 151
- intrinsics 11
- inverse query 307
- IP (TELNET subcommand) 118
- IUCV cross-memory facility 248

J

- JIS78KJ 64
- JIS83KJ 64

K

- Kanji
 - EUCKANJI 61
 - IBMKANJI 63
 - JIS78KJ 64
 - JIS83KJ 64
 - SJISKANJI 83
 - support for FTP 335
- Kanji, using 335
- KDESTROY (Kerberos subcommand) 166
- Kerberos 163, 169
- Kerberos Authentication System 11
 - commands
 - KDESTROY 166
 - KINIT 164
 - KLIST 165
 - KPASSWD 167
 - name structures 163
 - overview of Kerberos 11, 163
- KINIT (Kerberos subcommand) 164
- KLIST (Kerberos subcommand) 165
- KPASSWD (Kerberos subcommand) 167

L

- layered architecture
 - application layer 4
 - internetwork layer 4
 - network layer 4
 - transport layer 4
- LEVEL (NETSTAT parameter) 131
- licensing agreement 382
- line mode 116
- local address 14
- Local Area Network (LAN) 1
- local host 2
- LOCSTAT (FTP subcommand) 66
- LOCSTAT (TFTP subcommand) 99

- logical network 2
- LPQ
 - command 224
 - examples 225
 - usage 225
- LPQ Command 224
- LPR
 - command 212
 - examples 220
 - usage 218
- LPR Command 212
- LPRM
 - command 226
 - examples 227
 - usage 227
- LPRSET
 - command 207
 - examples 210
 - usage 209
- LPRSET Command 207
- LS (FTP subcommand) 67

M

- mail, electronic 105, 110
- Management Information Base (MIB) objects 341
- mapping 2
- mapping values (APL2) 329
- MAXPKT (TFTP subcommand) 99
- MDELETE (FTP subcommand) 69
- message examples, notation used in xiii
- MGET (FTP subcommand) 69
- MIB/Network elements
 - Address Translation group 350
 - ibm3172ifBlkCounters table 366
 - ibm3172ifChanCounters table 364
 - ibm3172ifDbkCounters table 367
 - ibm3172ifDevice table 368
 - ibm3172ifLANCounters table 365
 - ibm3172ifTrap table 363
 - ibm3172System table 362
 - ICMP group 357
 - Interfaces group 346
 - IP Address Translation table 356
 - IP group 351
 - System group 345
 - TCP group 359
 - UDP group 361
- MODE (FTP subcommand) 71
- MODE (TFTP subcommand) 100
- monitoring the TCP/IP network 121, 158
- MOUNT (NFS subcommand) 170
- MOUNTPW (NFS subcommand) 179
- MPUT (FTP subcommand) 71
- MVS data sets 323

N

- name translation, NFS 196
- NDB client machine 266
- NDB limitations 274

- NDB server 269
- NDB server machines 268
- NDBCLNT Command 270
- NEighbor (NETSTAT parameter) 131
- NETRC (FTP subcommand) 72
- NETRC DATA File
 - FTP 31
 - how to use 327
 - REXEC 203
- NETSTAT 121, 155
 - LEVEL 131
- NETSTAT address interpretation 121
- NETSTAT command format 123
- NETSTAT examples 135
- NETSTAT parameters
 - ALL 123
 - ALLCONN 124
 - ARP 124
 - BLOCK 124
 - CLIENTS 125
 - CONN 126
 - CP 126
 - DELARP 126
 - DELNeighbor 126
 - DEVLINKS 127
 - DROP 129
 - FILTERS 129
 - GATE 129
 - HELP 130
 - HOME 130
 - IDENTIFY 130
 - INTERVAL 130
 - NEighbor 131
 - POOLSIZE 132
 - RESETPOOL 133
 - SOCKET 133
 - TELNET 134
 - UNBLOCK 134
 - UP 134
- NetView Command List Language 247
- network
 - logical 2
 - physical 2
 - STATUS 121
- network address format 14
- Network Database System (NDB) 12, 265, 277
- network layer 4
- network management 247
- network number 14
- network protocols 6
- NFS 169, 198
- NFS CMS SMSG command 190
- NFS commands
 - Error Messages 170, 172, 183
 - MOUNT 170, 172
 - MOUNTPW 170, 172, 179
 - UMOUNT 170, 172
- NFS name translation
 - special file names 196
- NFS overview 12
- NFS PC-NFS client 197

- NFS servers, RACF support 198
- NFS system return codes 183
- NOOP (FTP subcommand) 73
- notation used in message and response examples xiii
- NSLOOKUP command 284, 285
- NSLOOKUP Command Line Queries 284
- NSLOOKUP Interactive Session Queries 285, 294
- NSLOOKUP internal state information 283
- NSLOOKUP options 290
- NSLOOKUP program 283

O

- OBEY (NETSTAT parameter) 132
- obtaining information from the network 121
- octet mode
 - with the TFTP GET subcommand 98
 - with the TFTP MODE subcommand 100
- OfficeVision 105
- OPEN (FTP subcommand) 73
- OPEN (TFTP subcommand) 100
- Open System Interconnection (OSI) 1
- OS/2 files 325
- OSF/Motif 11

P

- PA1 (TELNET subcommand) 119
- packet 2
- partitioned data set names 324
- PASS (FTP subcommand) 74
- PASSIVE (FTP subcommand) 75
- password use with FTP
 - ACCT 51
 - PASS 74
 - QUOTE 77
 - USER 88
- physical network 2
- PING 159
- PING command format 159
- PING options
 - COUNT 160
 - LENGTH 160
 - TIMEOUT 160
- Ping-o-Death attack 129
- POOLSIZE (NETSTAT parameter) 132, 152
- port 2
- Port Manager client 268
- Port Manager server 267, 268
- PROFS interface 109
- protocol data unit (PDU) 248
- protocols
 - definition 2
 - internetwork protocols 6
 - network protocols 6
- PUT (FTP subcommand) 75
- PUT (TFTP subcommand) 101
- PWD (FTP subcommand) 76

Q

- query engine 248
- query options 296
- query types
 - inverse query 307
 - standard query 307
- querying name servers
 - DiG 294
 - NSLOOKUP 283
- QUIT (FTP subcommand) 77
- QUIT (TELNET subcommand) 119
- QUIT (TFTP subcommand) 101
- QUOTE (FTP subcommand) 77

R

- RACF 96, 198
- raw socket 248
- related protocols 373
- Remote Execution Protocol (REXEC) 12, 201, 207
- remote printing 11, 207, 227
- Remote Procedure Call (RPC) 12, 265
- RENAME (FTP subcommand) 78
- RESETPOOL (NETSTAT parameter) 133, 153
- resolvers 279
- resource records 280
- response examples, notation used in xiii
- REXEC command format 201
- REXMIT (TFTP subcommand) 101
- RouteD overview 11
- router 2
- routing
 - direct 13
 - indirect 13
- RPC client 266, 267
- RPC server 268
- RPCINFO 155
- RPCINFO command format 156
- RPCINFO parameters 155

S

- Secure Socket Layer (SSL) 13
- security
 - database 274
 - system 274
- SENDPORT (FTP subcommand) 79
- SENDSITE (FTP subcommand) 80
- sequential data set names 324
- server 2
- SET subcommand 290
- Simple Mail Transfer Protocol (SMTP) 10
 - Using DBCS with Mail 339
- Simple Network Management Protocol (SNMP) 10
- SITE (FTP subcommand) 80
- SIZE (FTP subcommand) 83
- SJISKANJI 83
- SMSG (CMS subcommand) 190
- SMSG interface to SMTP 110
- SMTPQUEUE 106

- SMTPQUEUE Command 106
- Smurf attack 129
- SNMP
 - command 249
 - command processor 248
 - generic TRAP types 371
 - IBM 3172 enterprise-specific MIB variables 262
 - major and minor error codes 261
 - managing an internet environment 247
 - MIB/Network elements 344
 - overview 10
 - Protocol 247
 - Query Engine Host Lookup 262
 - return codes 249
 - sample command lists 247
 - value types
 - SNMP GET 250
 - SNMP GETNEXT 252
 - SNMP MIBVNAME 259
 - SNMP PING 260
 - SNMP SET 254
 - SNMP TRAPSOFF 258
 - SNMP TRAPSON 255
- SNMP command processor 248
- SNMP generic TRAP types 371
- SNMP Get command 252
- SNMP GET command 250
- SNMP MIBVNAME command 259, 260
- SNMP Set command 254
- SNMP TRAPSOFF command 258
- SNMP TRAPSON command 255
- SNMP/Netview overview 248
- SNMPIUCV task 248
- SNMPMGMT command 247
- SNMPRUN command 247
- SOCKET (NETSTAT parameter) 133, 153
- socket interfaces 13
- sockets
 - datagram 248
 - raw 248
 - stream 248
- SQL commands
 - imbedded 266
 - interactive 266
- SQL/DS 265
- standard query 307
- STATUS (FTP subcommand) 83
- stream socket 248
- STRUCT (FTP subcommand) 84
- Structure and Identification of Management Information (SMI) 341
- subnetwork address format 16
- subnetwork number 14
- SUNIQUE (FTP subcommand) 84
- SYNCH (TELNET subcommand) 119
- syntax diagrams, how to read xi
- SYSTEM (FTP subcommand) 86

T

- TCP/IP functional groups
 - application layer 4
 - internetwork layer 4
 - network layer 4
 - transport layer 4
- TCP/IP protocols and functions 4
- TELNET 115, 120
- TELNET (NETSTAT subcommand) 134, 154
- TELNET command format 115
- TELNET control characters in line mode 120
- TELNET function keys 116
- TELNET modes
 - line mode 117
 - transparent mode 117
- TELNET parameters
 - FIREWALL 115
 - LINEMODE 115
 - RECORD 116
 - TRANSLATE 116
- TELNET PF key functions 116
- Telnet Protocol 9
- TELNET subcommands
 - AO 117
 - AYT 117
 - BRK 118
 - HELP 118
 - IP 118
 - PA1 119
 - QUIT 119
 - SYNCH 119
- TELNET support display stations 116
- TFTP 97, 101
- TFTP command format 97
- TFTP parameters
 - ASCII 100
 - OCTET 100
 - TRANSLATE 97
- TFTP subcommands
 - GET 98
 - HELP 99
 - LOCSTAT 99
 - MAXPKT 99
 - MODE 100
 - OPEN 100
 - PUT 101
 - QUIT 101
 - REXMIT 101
 - TRACE 102
- Token-Ring 1, 2
- TRACE (TFTP subcommand) 102
- traceroute function (TRACERTE) 156
- translation tables
 - Chinese DBCS 320
 - converting to binary 320
 - Hangeul DBCS 319
 - IBM 314
 - Japanese Kanji DBCS 318
 - search order 312
- translation tables for TFTP 98
- Transmission Control Protocol (TCP) 8

- transparent mode 116
- transport layer 4
- transport protocols
 - Transmission Control Protocol (TCP) 8
 - User Datagram Protocol (UDP) 9
- Trivial File Transfer Protocol (TFTP) 9, 97
- TYPE (FTP subcommand) 87

U

- UNBLOCK (NETSTAT parameter) 134
- UNIX syntax 62
- UP (NETSTAT subcommand) 134, 155
- user 2
- USER (FTP subcommand) 88
- User Datagram Protocol (UDP) 9
- using translation tables 311

V

- virtual circuitry 287
- virtual network 1

W

- Wide Area Network (WAN) 1
- widgets
 - Athena 11
 - OSF/Motif 11
- Windows 95 files 325
- write logic for file mode 6 188

X

- X-Toolkit overview 11
- X-Windows overview 11
- X.25 6

Readers' Comments — We'd Like to Hear from You

z/VM
TCP/IP User's Guide
version 5 release 2

Publication No. SC24-6127-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name

Address

Company or Organization

Phone No.



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, New York 12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5741-A05

Printed in USA

SC24-6127-01



Spine information:



z/VM

TCP/IP User's Guide

version 5 release 2