

z/VM
7.3

TCP/IP Planning and Customization



Note:

Before you use this information and the product it supports, read the information in [“Notices” on page 725.](#)

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-29

© **Copyright International Business Machines Corporation 1987, 2025.**

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

Contents

Figures.....	xv
Tables.....	xvii
About This Document.....	xxi
Intended Audience.....	xxi
Conventions and Terminology.....	xxi
How the Term “internet” Is Used in This Document.....	xxi
Syntax, Message, and Response Conventions.....	xxi
Where to Find More Information.....	xxiv
Links to Other Documents and Websites.....	xxiv
How to provide feedback to IBM.....	xxv
Summary of Changes for z/VM: TCP/IP Planning and Customization.....	xxvii
SC24-6331-73, z/VM 7.3 (September 2025)	xxvii
SC24-6331-73, z/VM 7.3 (June 2025)	xxvii
SC24-6331-73, z/VM 7.3 (September 2023).....	xxviii
SC24-6331-73, z/VM 7.3 (September 2022).....	xxviii
Chapter 1. Planning Considerations.....	1
Introducing TCP/IP.....	1
Connectivity and Gateway Functions.....	1
Server Functions.....	1
Client Functions.....	3
Network Status and Management Functions.....	3
Application Programming Interfaces.....	3
Migration Information and Resources.....	4
User ID Privilege Class Considerations.....	4
User ID Minidisk Considerations.....	5
Shared File System (SFS) Considerations.....	5
Implications of Assigning Different Server Virtual Machine Names.....	6
Accommodating Changed Server Names.....	6
Multiple Server Instance Restrictions.....	9
Mutually Exclusive Servers.....	9
Publication References.....	9
Chapter 2. System Requirements for TCP/IP.....	11
z/VM Device Definition Considerations.....	11
Hardware Environment.....	11
Network Attachments.....	11
Open System Adapter-Express (OSA-Express).....	11
HiperSockets.....	11
Channel-to-Channel Support.....	11
IUCV.....	12
z/VM Virtual Network Adapters.....	12
Software Environment.....	12
Chapter 3. Defining the TCP/IP System Parameters.....	13

Configuring the TCPIP DATA File.....	13
Statement Syntax.....	13
ATSIGN statement.....	14
DOMAINLOOKUP statement.....	14
DOMAINORIGIN statement.....	15
DOMAINSEARCH statement	16
HOSTNAME statement.....	18
HOSTVERIFICATION statement.....	18
NSINTERADDR statement.....	19
NSPORTADDR statement.....	20
RESOLVERTIMEOUT statement.....	20
RESOLVERUDPRETRIES statement.....	20
RESOLVEVIA statement.....	21
SECURETELNETCLIENT statement.....	21
SMTPSERVERID statement.....	21
TCPIPUSERID statement.....	22
TRACE RESOLVER statement.....	22
UFTSERVERID statement.....	23
USERDATA statement.....	23
VMFILETYPE statement.....	23
VMFILETYPEDEFAULT statement.....	24
Testing the TCP/IP System Configuration.....	25
HOMETEST Command.....	25
Chapter 4. Configuring the Local Host Files.....	27
ETC HOSTS File Syntax.....	27
HOSTS LOCAL File Syntax.....	28
HOST Statement.....	28
NET Statement.....	29
Building the HOSTS LOCAL Site Table.....	30
Chapter 5. General TCP/IP Server Configuration.....	33
Virtual Machine Definitions.....	33
Required Virtual Machines.....	33
Optional Virtual Machines.....	33
Methods of Server Configuration.....	35
The DTCPARMS File.....	35
Configuring the DTCPARMS File.....	36
Customizing Servers.....	43
Automatic Generation of Selected Startup Parameters.....	44
Adding New Servers and Server Classes.....	44
Duplicating and Running Existing Servers.....	44
Server Profile Exits.....	46
Global Profile Exit.....	48
IBM Diagnostic Profile Exit.....	48
Customizing Server-specific Exits.....	49
GCS Servers.....	49
TCP/IP Configuration File Overview.....	50
Server Administrative Command Interface Summary.....	51
Stopping TCP/IP Servers.....	51
Starting TCP/IP Servers.....	52
TCP/IP and SSL Server Logon Restrictions.....	52
Chapter 6. Configuring the FTP Server.....	55
Step 1: Update PROFILE TCPIP.....	55
Step 2: Update the DTCPARMS File.....	55
SRVRFTP Command Operands (:Parms. Parameters).....	56

SRVRFTP Command Syntax.....	56
Step 3: Establish FTP Server Machine Authorizations.....	56
Step 4: Customize the FTP Server Configuration File.....	57
FTP Server Configuration File Statements.....	57
ANONYMOU Statement.....	57
AUTOTRANS Statement.....	58
CLIENTCERTCHECK Statement.....	58
DONTREDIRECT Statement.....	59
FTAUDIT Statement.....	59
FTCHKCMD Statement.....	60
FTCHKDIR Statement.....	60
FTPKEEPALIVE Statement.....	60
INACTIVE Statement.....	61
LISTFORMAT Statement.....	61
LOADDBCSTABLE Statement.....	62
PASSIVEPORTRANGE Statement.....	63
PORT Statement.....	63
RACF Statement.....	63
RDR Statement.....	64
RESTRICTUSE Statement.....	65
SECURECONTROL Statement.....	65
SECUREDATA Statement.....	66
SYSTEMGREETING Statement.....	66
TIMESTAMP Statement.....	67
TSLABEL Statement.....	67
TRACE Statement.....	68
Step 5: Configure Automatic File Translation (Optional).....	68
Step 6: Configure Secure FTP Connections (Optional).....	69
Step 7: Customize FTP Server Exits (Optional).....	69
Using the FTP Welcome Banner.....	69
Using the FTP Server Exit.....	70
Using the CHKIPADR Exit.....	70
CHKIPADR Input.....	72
CHKIPADR Output.....	72
Example.....	72
Dynamic Server Operation.....	73
SMSG Interface to the FTP Server.....	74
Providing Web Browser FTP Support.....	78

Chapter 7. Configuring the LDAP Server.....79

Configuration Steps for the LDAP Server.....	79
Step 1: Update the TCP/IP Server Configuration File (PROFILE TCPIP).....	79
Step 2: Update the DTCPARMS File for the LDAP Server.....	80
Step 3. Determine the LDAP Server BFS Directory Default.....	81
Step 4. Set Up the User ID and Security for the LDAP Server.....	82
Step 5. Copy the Configuration Files.....	83
Step 6. Create and Customize the LDAP Configuration File (DS CONF).....	83
Step 7. Set the Time Zone.....	105
Step 8. Set Environment Variables (DS ENVVARS).....	105
Step 9. Verify the LDAP Server.....	109
Step 10. Finalize Setup of LDAP Backends.....	110
Setting up for SDBM.....	111
Setting up for GDBM.....	112
Setting up for CDBM.....	113
Configuring remote services support.....	114
Setting up for SSL/TLS.....	114
Using SSL/TLS Protected Communications.....	114

Enabling the LDAP Server to Use IBM Z Cryptographic Hardware.....	115
Creating and Using a Key Database.....	116
Obtaining a Certificate.....	116
Enabling SSL/TLS Support.....	116
Setting up the Security Options for the LDAP Server.....	116
Setting up an LDAP Client.....	121
Support of Certificate Bind.....	121
Configuring for Encryption or Hashing.....	121
One-way Hashing Formats.....	122
Two-way Encryption Formats.....	123
Symmetric Encryption Keys.....	123
Configuring for user and administrator password encryption or hashing.....	123
Configuring for Secret Encryption.....	124
Configuring Plug-in Extensions.....	125
Example Configuration Scenarios.....	125
Configuring SDBM and LDBM Backends.....	125
Configuring LDBM with Native Authentication and GDBM Backends.....	126
Configuring RACF/VM Change Logging with SDBM and GDBM Backends.....	127
Configuration File (DS CONF) Format and Configuration Options.....	127
Specifying a Value for Filename.....	129
Specifying a Value for a Distinguished Name.....	129
Configuration File Checklist	130
Configuration File Options	132
Dynamic Server Operation.....	161
SMSG Interface to the LDAP Server.....	161
Dynamic Debugging.....	163
Activity logging.....	163
LDAP SMF Auditing.....	168
Monitoring LDAP Server Resources.....	170
Running and Using the LDAP Backend Utilities.....	171
Running the Backend Utilities in CMS.....	171
SSL/TLS Information for LDAP Utilities.....	171
DB2PWDEN (db2pwdcn utility).....	172
DS2LDIF (ds2ldif utility).....	175
LDAPEXOP (ldapexop utility).....	181
Internationalization Support	190
Translated Messages.....	190
UTF-8 Support.....	190

Chapter 8. Configuring the MPRoute Server..... 193

Understanding MPRoute.....	193
Dynamic routing.....	194
IPv4 dynamic routing using MPRoute.....	194
IPv6 dynamic routing using MPRoute.....	196
Using RIP, IPv6 RIP, OSPF, and IPv6 OSPF with MPRoute.....	197
Preventing futile neighbor state loops during adjacency formation.....	198
Special considerations.....	199
Dynamic Server Operation.....	200
Configuration Steps for the MPRoute Server.....	200
Step 1. Update the TCP/IP server configuration file.....	200
Step 2. Update the ETC SERVICES file.....	201
Step 3. Create the MPRoute Configuration File.....	201
Step 4. Optional: Update the DTCPARMS File.....	201
Step 5. Optional: Create static routes.....	202
Step 6. Optional: Configure OSPF authentication if using the IPv4 OSPF protocol.....	202
MPROUTE Command.....	203
MPRoute configuration file.....	203

INCLUDE.....	204
Creating the MPRoute configuration file.....	205
OSPF configuration statements.....	218
RIP configuration statements.....	232
IPv6 OSPF configuration statements.....	241
IPv6 RIP configuration statements.....	250
Common configuration statements for RIP and OSPF.....	257
Dynamic Server Operation.....	261
SMSG Interface to the MPRoute Server.....	261
Chapter 9. Configuring the NFS Server.....	325
Step 1: Update PROFILE TCPIP.....	325
Step 2: Update the DTCPARMS File.....	325
VMNFS Command Operands (:Parms. Parameters).....	326
VMNFS Command Syntax.....	326
Using an External Security Manager.....	327
Step 3: Establish NFS Server Machine Authorizations.....	327
Step 4: Customize the VMNFS CONFIG File.....	327
NFS Configuration File Statements.....	327
Syntax Rules.....	328
DUMPMOUNT Statement.....	328
EXPORT Statement.....	329
EXPORTONLY Statement.....	330
MAXTCPUSERS Statement.....	330
PCNFSD Statement.....	331
VMFILETYPE Statement.....	332
Step 5: Configure NFS Server File Translation Support (Optional).....	332
Step 6: Verify NFS Server Operations.....	332
Step 7: Advanced Configuration Considerations.....	334
NFS Server Exits.....	334
Managing Translation Tables.....	338
Allowing Access to Migrated SFS and BFS Files.....	339
Managing Data Transfer Operations.....	339
Managing File Handle Operations.....	340
Using Additional Security Capabilities.....	341
Dynamic Server Operation.....	342
SMSG Interface to the NFS Server.....	342
SMSG CMS Command.....	342
SMSG REFRESH CONFIG Command.....	343
SMSG TWRITE Command.....	344
Chapter 10. Configuring the Portmapper Server.....	347
Step 1: Update PROFILE TCPIP.....	347
Step 2: Update the DTCPARMS File.....	347
PORTMAP Command Operands (:Parms. Parameters).....	347
PORTMAP Command Syntax.....	348
Step 3: Verify Portmapper Services.....	348
Chapter 11. Configuring the REXEC Server.....	349
Step 1: Update PROFILE TCPIP.....	349
Step 2: Update the DTCPARMS File.....	349
REXECD Command Operands (:Parms. Parameters).....	350
REXECD Command Syntax.....	350
Step 3: Define Additional Anonymous REXEC Agent Virtual Machines (Optional)	350
Step 4: Establish REXEC Server Machine Authorizations.....	351
Using an External Security Manager.....	351
Additional REXEC Considerations.....	351

How the REXEC Server Uses Secondary Virtual Machines.....	351
Anonymous REXEC Client Processing.....	351
User's Own Virtual Machines.....	352
Usage Notes.....	352
Chapter 12. Configuring the RSCS Print Server.....	353
Configuring a TN3270E Printer.....	353
Configuring an RSCS LPR Link.....	353
RSCSTCP CONFIG Configuration File.....	353
Configuring a Non-PostScript Printer.....	354
Available EPARMS for Non-PostScript Printers.....	355
Configuring a PostScript Printer.....	356
Available EPARMS for PostScript Printers.....	357
Configuring an RSCS LPD Link.....	361
Available EPARMS for LPD Links.....	363
Configuring an RSCS TN3270E Printer Link.....	365
TAG Command for a TN3270E printer.....	371
Chapter 13. Configuring the SMTP Server.....	375
Step 1: Update PROFILE TCPIP.....	375
Step 2: Update the System (CP) Directory for the SMTP Server.....	375
Step 3: Update the DTCPARMS File.....	375
SMTP Command Operands (:Parms. Parameters).....	376
SMTP Command Syntax.....	376
Step 4: Update the TCPIP DATA File for Domain Name Resolution.....	376
Step 5: Customize the SMTP CONFIG File.....	377
Step 6: Additional SMTP Server Considerations.....	377
Use of MX Records.....	377
Local versus Non-local Mail Recipients.....	377
SMTP Server Configuration File Statements.....	378
ALTRSCSDOMAIN Statement.....	381
ALTTCPHOSTNAME Statement.....	381
BADSPoolFILEID Statement.....	381
DBCS Statement.....	381
FILESPerCONN Statement.....	383
FINISHOPEN Statement.....	383
FORWARDMAIL Statement.....	383
GATEWAY Statement.....	384
INACTIVE Statement.....	385
IPMAILERADDRESS Statement.....	385
LOCALFORMAT Statement.....	386
LOG Statement.....	387
MAILER Statement.....	387
MAILHOPCOUNT Statement.....	388
MAXCONNPerSITE Statement.....	389
MAXMAILBYTES Statement.....	389
NOLOG Statement.....	389
ONDISKFULL Statement.....	390
OUTBOUNDOPENLIMIT Statement.....	391
PORT Statement.....	391
POSTMASTER Statement.....	391
RCPTRESPONSEDELAY Statement.....	392
RESOLVERRETRYINT Statement.....	393
RESTRICT Statement.....	393
RETRYAGE Statement.....	394
RETRYINT Statement.....	394
REWRITE822HEADER Statement.....	394

RSCSDOMAIN Statement.....	395
RSCSFORMAT Statement.....	395
SECURE Statement.....	396
MSGAUTHLIST Statement.....	396
SMTPCMD5 Statement.....	397
SOURCEROUTES Statement.....	399
SUPPRESSNOTIFICATION Statement.....	400
TEMPERRORRETRIES Statement.....	400
TLS Statement.....	401
TLSLABEL Statement.....	402
TRACE Statement.....	402
VERIFYBATCHSMTPSENDER Statement.....	403
VERIFYCLIENT Statement.....	404
VERIFYCLIENTDELAY Statement.....	405
WARNINGAGE Statement.....	405
8BITMIME Statement.....	406
Configuring the Server for Secure SMTP.....	406
SMTP Server Exits.....	407
Configuring a TCP/IP-to-RSCS Mail Gateway.....	407
SMTPRSCS Command.....	408
Configuring a TCP/IP-to-RSCS Secure Mail Gateway.....	409
Creating an SMTP Security Table.....	409
Operands.....	409
Defining Nicknames and Mailing Lists Using the SMTP NAMES File.....	411
Customizing SMTP Mail Headers.....	412
The SMTP RULES File.....	413
Format of the Field Definition Section.....	413
Format of the Rule Definition Section.....	415
Syntax Convention of the SMTP Rules.....	415
Predefined Keywords within the SMTP Rules.....	417
Default SMTP Rules.....	418
SMTP Non-Secure Gateway Configuration Defaults.....	418
SMTP Secure Gateway Configuration Defaults.....	418
Examples of Header Rewrite Rules.....	419
Dynamic Server Operation: MSG Interface to the SMTP Server.....	420
Privileged User MSG Commands.....	420

Chapter 14. Configuring the SNMP Servers..... 439

SNMP Overview.....	439
Configuring the SNMP Daemon.....	439
Step 1: Update PROFILE TCPIP.....	439
Step 2: Update the DTCPARMS File for SNMPD and SNMPSUBA.....	440
Step 3: Create the MIB Data File.....	441
Step 4: Configure the SNMP Daemon.....	441
SNMPD Command.....	441
TRAP Destination file.....	442
PW SRC File.....	442
SNMP Daemon Installation Steps.....	444
SNMP Daemon.....	444
Setting up an SNMP Subagent.....	444
Adding User-defined MIBs to an SNMP Subagent.....	445
Configuring the SNMP Client.....	445
SNMP Client Overview.....	445
Step 1: Update PROFILE TCPIP.....	447
Step 2: Update the DTCPARMS File for SNMPQE.....	447
SQESERV Command Operands (:Parms. Parameters).....	447
SQESERV Command Syntax.....	447

Step 3: Create the MIB Data File.....	448
Step 4: Configure the SNMP/NetView Interface.....	449
SNMPIUCV.....	449
SNMP Command Processor.....	449
SNMP Messages.....	450
SNMPIUCV Initialization Parameters.....	450
SNMP Client Installation Steps.....	451
SNMP Command Processor and SNMPICUV on NetView.....	451
Chapter 15. Configuring the SSL Server.....	453
Overview of an SSL Session.....	454
Understanding Certification Validation.....	455
Certification Authorities and Self-Signed Certificates.....	456
SSL/TLS Partner Certificate Revocation Checking.....	457
Enabling OCSP Support.....	457
Enabling HTTP CDP Support.....	458
The :OCSPParms. Tag.....	459
Step 1: Determine the SSL Server Configuration For Your Installation.....	462
Single Versus Multiple SSL Server Configurations.....	462
Step 1a: Enabling the SSL Server to Use IBM Z Cryptographic Hardware.....	465
Step 2: Update the TCP/IP Server Configuration File (PROFILE TCPIP).....	466
Step 3: Update the DTCPARMS File for the TCP/IP Server.....	466
Step 4: Update the DTCPARMS File for the SSL DCSS Management Agent Server.....	467
SSLIDCSS Command.....	467
Step 5: Update the DTCPARMS File for the SSL Server Pool.....	468
VMSSL Command Operands (:Parms. Parameters).....	469
VMSSL Command Syntax.....	470
Step 6: Set Up the Certificate (Key) Database.....	480
Step 7: Implement Customization for Protected Communications.....	484
Step 7A. Designate the Secure Ports (Static SSL Connections).....	484
Step 7B. Configure TLS Services (Dynamic SSL/TLS Connections).....	485
Dynamic Server Operation.....	485
SSL Server Administration.....	485
SSL Server Administration Commands.....	487
General SSLADMIN Command.....	488
SSLADMIN CLEAR Command.....	490
SSLADMIN CLOSECON Command.....	490
SSLADMIN HELP Command.....	490
SSLADMIN LOG Command.....	490
SSLADMIN QUERY Command.....	491
SSLADMIN REFRESH Command.....	497
SSLADMIN RESTART Command.....	497
SSLADMIN SET Command.....	497
SSLADMIN START Command.....	498
SSLADMIN STOP Command.....	498
SSLADMIN SYSTEM Command.....	498
SSLADMIN TRACE/NOTRACE Command.....	500
SSLPOOL Command.....	502
Migrating Certificates From a Prior-Level SSL Server Certificate Database.....	504
Chapter 16. Configuring the TCP/IP Server.....	507
TCPIP Virtual Machine Configuration Process.....	507
Step 1: Create a Multiprocessor Configuration.....	507
Step 2: Update the DTCPARMS File.....	508
Step 3: Create an Initial Configuration File.....	508
TCP/IP Configuration Statements.....	523
Summary of TCP/IP Configuration Statements.....	524

ACBPOOLSIZE Statement.....	527
ADDRESSTRANSLATIONPOOLSIZE Statement.....	528
ARPAGE Statement.....	529
ASSORTEDPARMS Statement.....	529
AUTOLOG Statement.....	534
BLOCK Statement.....	535
CCBPOOLSIZE Statement.....	536
DATABUFFERLIMITS Statement.....	537
DATABUFFERPOOLSIZE Statement.....	537
DEVICE and LINK Statements.....	539
Intelligent Default MTU Values Based on the Device and Link Type.....	539
DEVICE and LINK statements for CTC devices.....	539
DEVICE and LINK statements for HiperSockets connections.....	542
DEVICE and LINK statements for local IUCV connections.....	545
DEVICE and LINK statements for remote IUCV connections.....	548
DEVICE and LINK statements for LCS devices.....	550
DEVICE and LINK statements for OSD devices.....	553
DEVICE and LINK statements for OSH devices.....	559
DEVICE and LINK statements for virtual IP addressing (VIPA) devices.....	563
ENVELOPEPOOLSIZE Statement.....	564
FILE Statement.....	565
FIXEDPAGESTORAGEPOOL.....	566
FOREIGNIPCONLIMIT Statement.....	567
FOREIGNIPPOOLSIZE Statement.....	568
GATEWAY Statement.....	568
HOME Statement.....	579
ICMPERRORLIMIT Statement.....	583
INFORM Statement.....	584
INTERNALCLIENTPARMS Statement.....	584
IPROUTEPOOLSIZE Statement.....	590
KEEPALIVEOPTIONS Statement.....	590
LARGEENVELOPEPOOLSIZE Statement.....	591
LESSTRACE Statement.....	592
MAXRESTART Statement.....	593
MONITORRECORDS Statement.....	593
MORETRACE Statement.....	595
NCBPOOLSIZE Statement.....	595
NOSCREEN Statement.....	596
NOTRACE Statement.....	597
OBEY Statement.....	597
PACKETTRACESIZE Statement.....	598
PATHMTUAGE Statement.....	600
PENDINGCONNECTIONLIMIT Statement.....	600
PERMIT Statement.....	601
PERSISTCONNECTIONLIMIT Statement.....	602
PORT Statement.....	603
PRIMARYINTERFACE Statement.....	606
RCBPOOLSIZE Statement.....	608
RESTRICT Statement.....	608
ROUTERADV Statement.....	609
ROUTERADVPREFIX Statement.....	611
SCBPOOLSIZE Statement.....	613
SCREEN Statement.....	614
SKCBPOOLSIZE Statement.....	614
SMALLDATABUFFERPOOLSIZE Statement.....	615
SOMAXCONN Statement.....	616
SSLLIMITS Statement.....	616
SSLSERVERID Statement.....	617

START Statement.....	618
STOP Statement.....	618
SYSCONTACT Statement.....	619
SYSLOCATION Statement.....	619
TCBPOOLSIZE Statement.....	620
TIMESTAMP Statement.....	621
TINYDATABUFFERPOOLSIZE Statement.....	621
TN3270E Statement.....	622
TRACE Statement.....	623
TRACEONLY Statement.....	625
TRANSLATE Statement.....	626
UCBPOOLSIZE Statement.....	627
UDPQUEUELIMIT Statement.....	627
VSWITCH CONTROLLER Statement.....	628
Changing the TCP/IP Configuration with the IFCONFIG Command.....	631
IFCONFIG Command.....	631
Changing the TCP/IP Configuration with the OBEYFILE Command.....	644
OBEYFILE Command.....	644
Starting and Stopping TCP/IP Services.....	646
Chapter 17. Configuring the UFT Server.....	649
Step 1: Update PROFILE TCPIP.....	649
Step 2: Update the DTCPARMS File.....	649
UFTD Command.....	649
Step 3: Update the TCPIP DATA File.....	650
Step 4: Customize the UFTD CONFIG File.....	650
UFT Configuration File Statements.....	650
IDENTIFY Statement.....	650
MAXFILEBYTES Statement.....	651
NSLOOKUP Statement.....	651
PORT Statement.....	652
TRACE Statement.....	652
TRANSLATE Statement.....	653
UFTCMDS EXIT Statement.....	653
Step 5: Advanced Configuration Considerations.....	655
DNS Lookup Exit.....	655
Protocol Commands Exit.....	655
Dynamic Server Operation.....	656
UFTD Subcommands.....	657
IDENTIFY Subcommand.....	657
NSLOOKUP Subcommand.....	658
QUERY Subcommand.....	658
QUIT Subcommand.....	659
STOP Subcommand.....	659
TRACE Subcommand.....	660
UFTCMDS EXIT Subcommand.....	661
UFT Clients and Servers for Other Platforms.....	662
Chapter 18. Configuring the RSCS UFT Client.....	663
Step 1: Update the RSCSTCP CONFIG Configuration File.....	663
UFT Client LINKDEFINE and PARM Statements.....	663
Operands.....	664
Step 2: Update the RSCSUFT CONFIG Configuration File.....	664
Step 3: Update the TCPIP DATA File.....	664
Chapter 19. Using Translation Tables.....	667
Character Sets and Code Pages.....	667

TCP/IP Translation Table Files.....	667
Translation Table Search Order.....	668
Special Telnet Requirements.....	669
IBM-Supplied Translation Tables.....	669
Customizing SBCS Translation Tables.....	672
Syntax Rules for SBCS Translation Tables.....	672
Customizing DBCS Translation Tables.....	673
DBCS Translation Table.....	673
Syntax Rules for DBCS Translation Tables.....	673
Sample DBCS Translation Tables.....	674
Converting Translation Tables to Binary.....	675
Chapter 20. Testing and Verification.....	677
Loopback Testing.....	677
TCP/IP Checksum Testing.....	677
CHECKSUM Statement.....	677
NOCHECKSUM Statement.....	677
Chapter 21. Using Source Code Libraries.....	679
VMFASM EXEC, VMFHASM EXEC, and VMFHLASM EXEC.....	679
VMFPAS EXEC.....	679
VMFC EXEC.....	680
TCPTXT EXEC.....	681
TCPLOAD EXEC.....	681
TCPCOMP EXEC.....	682
Special Considerations.....	683
Appendix A. Using TCP/IP with an External Security Manager.....	685
Server Validation Methods.....	685
Security Interfaces.....	685
Server Initialization.....	686
Client Authentication.....	686
Resource Access.....	687
The DTCPARMS File.....	687
Minidisk Security.....	688
Using TCP/IP with RACF.....	688
Steps for using TCP/IP with RACF.....	688
Appendix B. SMF records.....	691
SMF Record Type 83, subtype 3 records.....	691
RACF SMF unload utility output.....	694
Appendix C. Activity Log Records.....	707
Activity Log Start and End Field Descriptions.....	707
Activity Log mergedRecord Field Descriptions.....	711
Appendix D. Related Protocol Specifications.....	715
Appendix E. Abbreviations and Acronyms.....	721
Notices.....	725
Programming Interface Information.....	726
Trademarks.....	726
Terms and Conditions for Product Documentation.....	727
IBM Online Privacy Statement.....	727

Bibliography.....	729
Where to Get z/VM Information.....	729
z/VM Base Library.....	729
z/VM Facilities and Features.....	730
Prerequisite Products.....	732
Related Products.....	732
Other TCP/IP Related Publications.....	732
 Index.....	 735

Figures

1. Native authentication example.....	96
2. General format of DS CONF.....	128
3. The SMTP Virtual Machine Configured as a Mail Gateway.....	407
4. Overview of NetView SNMP Support.....	446
5. Sample MIB_DESC DATA Line.....	449
6. Host routing under single subnet.....	510
7. Subnet assignment for destinations beyond a single hop.....	510
8. Basic host routing configuration.....	511
9. Adding hosts to subnetted interfaces.....	511
10. Single VIPA Configuration.....	515
11. Point-to-Point Link.....	521
12. Example of route types.....	571
13. Example of Network Connectivity Using Variable Subnetting.....	572
14. Example of Network Using equal-cost multipath routes.....	574
15. Intranet with Two Guest LANs.....	575
16. An IPv6 multicast default route on the GATEWAY statement.....	579

Tables

1. Examples of Syntax Diagram Conventions.....	xxii
2. TCP/IP Server and User ID Assigned Privilege Classes.....	4
3. Required TCP/IP Server Minidisk Links.....	33
4. Required Virtual Machines.....	33
5. Optional Virtual Machines.....	34
6. DTCPARMS File Search.....	35
7. DTCPARMS Tags for Configuring Servers.....	37
8. Server Parameters Generated at Initialization.....	44
9. TCP/IP Server-specific Exits.....	49
10. Configuration Files and Minidisk Location Summary.....	50
11. Operating modes for native authentication binding.....	89
12. The errno values returned by __passwd() when binding.....	90
13. Operating modes for updating native password or password phrases.....	91
14. The errno values returned by __passwd() when updating password or password phrase.....	93
15. Behavior of native authentication in example 1.....	96
16. Behavior of native authentication in example 2.....	97
17. cn=configuration entry attribute descriptions.....	100
18. cn=Replication,cn=configuration entry attribute descriptions.....	102
19. cn=Replication,cn=Log Management,cn=Configuration entry attribute descriptions.....	103
20. cn=safadmingroup,cn=configuration entry attribute descriptions.....	104
21. Debug levels.....	108
22. SSL ciphers supported by the sslCipherSpecs configuration option.....	118
23. Sample checklist and DS CONF (using SDBM and LDBM).....	126

24. Sample checklist and DS CONF (using GDBM and LDBM).....	126
25. Sample checklist and DS CONF (using SDBM and GDBM).....	127
26. Configuration file options checklist.....	130
27. Mapping between Unicode and UTF-8.....	191
28. Multipath route limitations.....	199
29. Route precedence.....	217
30. MPROUTE IPv4 Route Type and COST Value mapping.....	293
31. MPROUTE IPv6 Route Type and COST Value mapping.....	320
32. Correct Combinations for TRANS and FEATURE Settings.....	371
33. SMTP CONFIG Configuration Statements.....	378
34. Privileged SMTP SMSG Commands.....	420
35. Mail Forwarding Exit - Sample Queries.....	423
36. SMTP Command Exit - Sample Queries.....	431
37. Client Verification Exit - Sample Queries.....	437
38. SSLV2 Cipher Suite Values.....	474
39. SSLV3 and TLS Cipher Suite Values.....	475
40. SSL Administration Commands.....	487
41. SSLPOOL Command MAXPERSSLSERVER Values.....	504
42. Source VIPA Usage Chart.....	517
43. Free Pool Configuration Statements.....	522
44. Summary of TCP/IP Configuration Statements.....	524
45. Relationship of ASSORTEDPARMS options to IP protocols.....	533
46. TCP/IP Process Names.....	624
47. UFTD Subcommands.....	657
48. Translation Table Files.....	668

49. Preferred Translation Tables.....	668
50. IBM Translation Tables.....	670
51. Server validation methods.....	685
52. LDAP event codes.....	691
53. LDAP extended relocates.....	692
54. Event type strings.....	694
55. Event qualifiers.....	695
56. Event specific fields for LDAP add event (Event code 1).....	695
57. Event specific fields for LDAP bind event (Event code 2).....	696
58. Event specific fields for LDAP compare event (Event code 3).....	698
59. Event specific fields for LDAP connect, delete, disconnect, and unbind events (Event code 4, 5, 6, 11).....	699
60. Event specific fields for LDAP extended operations event (Event code 7).....	700
61. Event specific fields for LDAP modify event (Event code 8).....	701
62. Event specific fields for LDAP modify DN event (Event code 9).....	702
63. Event specific fields for LDAP search event (Event code 10).....	704
64. Start or end activity log fields.....	707
65. mergedRecord activity log fields.....	711

About This Document

z/VM: TCP/IP Planning and Customization describes how to plan the installation and perform the configuration of the IBM Transmission Control Protocol/Internet Protocol (function level 730) for z/VM.

This document describes how to define and configure the virtual machines, servers, and applications available in TCP/IP. This document also describes how to customize and tune TCP/IP for your specific needs.

This document describes all applications available with function level 730; however, your organization may use only some of these functions.

Intended Audience

This document is intended for system administrators to help in planning for TCP/IP networks on a z/VM host, and in customizing TCP/IP to their systems.

This document assumes that you are familiar with z/VM and its components, Control Program (CP) and the Conversational Monitor System (CMS).

Conventions and Terminology

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

How the Term “internet” Is Used in This Document

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

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

Syntax, Message, and Response Conventions

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

How to Read Syntax Diagrams

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

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

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

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

Table 1. Examples of Syntax Diagram Conventions

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

Table 1. Examples of Syntax Diagram Conventions (continued)

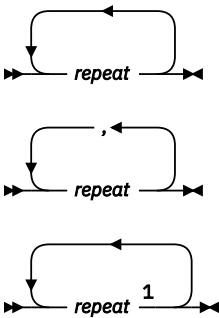
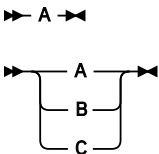
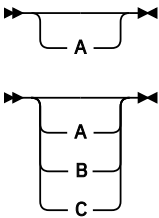
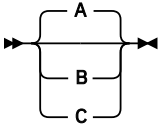
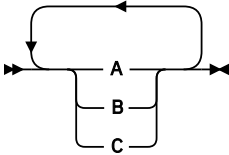
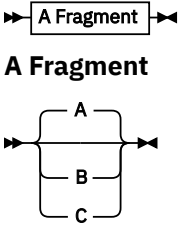
Syntax Diagram Convention	Example
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram.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

Table 1. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 The diagram shows two examples of syntax fragments. The first is a box labeled 'A Fragment' with arrows on either side. The second is a heading 'A Fragment' followed by a diagram where three sub-elements, labeled A, B, and C, are grouped by a large right-facing curly bracket. Arrows point to the start and end of this group.

Examples of Messages and Responses

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

xxx

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

[]

Brackets enclose optional text that might be displayed.

{ }

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

|

The vertical bar separates items within brackets or braces.

...

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

Where to Find More Information

Appendix E, "Abbreviations and Acronyms," on page 721, lists the abbreviations and acronyms that are used throughout this document.

For more information about related publications, see the documents listed in the "Bibliography" on page 729.

Links to Other Documents and Websites

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

How to provide feedback to IBM

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

Summary of Changes for z/VM: TCP/IP Planning and Customization

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

SC24-6331-73, z/VM 7.3 (September 2025)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.3.

The following topics are updated:

- “[7.3 PH65377, PH67660] z/VM TCP/IP support for enhanced QDIO” on page xxvii
- “IFCONFIG Command” on page 631

SC24-6331-73, z/VM 7.3 (June 2025)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.3.

| [7.3 PH65377, PH67660] z/VM TCP/IP support for enhanced QDIO

| With the PTFs for APARs PH65377 and PH67660, z/VM 7.3 provides a native network device driver for the z/VM TCP/IP stack that uses enhanced QDIO (EQDIO) adapters for network transport.

| **Note:** The EQDIO device driver requires the TCP/IP user ID to be configured to IPL ZCMS by default after the IBM z17 support has been applied.

The following topics are new:

- “Network Express adapter support” on page 519
- “DEVICE and LINK statements for OSH devices” on page 559
- “DEVICE Statement — OSH devices” on page 559
- “LINK statement for EQDIOETHERNET” on page 560

The following topics are updated:

- “Attaching network devices to the TCPIP virtual machine” on page 519
- “Intelligent Default MTU Values Based on the Device and Link Type” on page 539
- “DEVICE and LINK statements for OSD devices” on page 553
- “TRACE Statement” on page 623
- “IFCONFIG Command” on page 631

Miscellaneous updates for June 2025

The following topics are updated:

- “LINK Statement — CTC Devices” on page 540
- “LINK Statement — Local IUCV Connections” on page 546
- “LINK Statement — Remote IUCV Connections” on page 549

- [“LINK Statement for Ethernet Network LCS” on page 551](#)
- [“LINK Statement for QDIOETHERNET” on page 556](#)

SC24-6331-73, z/VM 7.3 (September 2023)

This edition supports product changes that were provided or announced after the general availability of z/VM 7.3.

[PH56199, VM66698] System SSL z/OS 2.5 equivalence

With the PTFs for APARs PH56199 (TCP/IP) and VM66698 (LE), z/VM 7.3 provides an update to the cryptographic services library, which includes certificate diagnostic enhancements and improved algorithmic support and allows for enablement of TLS 1.3, for secure connectivity to the z/VM platform.

SC24-6331-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Miscellaneous updates for z/VM 7.3

The following topic is new:

- [“UDPQUEUELIMIT Statement” on page 627](#)

The following topics are updated:

- [“Managing Data Transfer Operations” on page 339](#)
- [“VMSSL Command Syntax” on page 470](#)
- [“ASSORTEDPARMS Statement” on page 529](#)
- [“FOREIGNIPCONLIMIT Statement” on page 567](#)

Chapter 1. Planning Considerations

This chapter provides an introduction to the Transmission Control Protocol/Internet Protocol, TCP/IP, and describes the planning and preparation that you should consider before function level 730 is on your system.

Introducing TCP/IP

Transmission Control Protocol/Internet Protocol can be characterized as belonging to one of the following categories:

- Connectivity and gateway functions, which handle the physical interfaces and routing of data.
- Server functions, which provide a service to a client (that is, send or transfer a file).
- Client functions, which request a certain service from a server anywhere in the network.
- Network status/management functions, which detect and solve network problems.
- Application Programming Interfaces, which allow you to write your own client/server applications.

TCP/IP is used to build an interconnection between networks (or internet) through universal communication services. To be able to communicate between networks, addresses are assigned to each host on the network. This is called the **IP address**. Using an IP address, TCP/IP protocols can communicate between networks and hosts. For example, 9.76.22.1 is an IP address. Each IP address can also have an associated **domain name** or nickname. The domain name is an alternative method of referring to an IP address. For example the domain name VMHOST.RALEIGH.IBM.COM could be used to refer to the IP address 9.76.22.1.

Currently, there are two versions of IP protocols: Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6). IPv4 is a set of protocols that most networks use. IPv6 is a new generation of protocols designed to solve problems in IPv4. For a general introduction to TCP/IP for z/VM, including its support for IPv6, see *z/VM: TCP/IP User's Guide*. Also available is a description of TCP/IP IPv4 and IPv6 in *TCP/IP Tutorial and Technical Overview*, GG24-3376.

Connectivity and Gateway Functions

The following are connectivity and gateway functions that may be used to communicate with other networks either internal or external to your network. Each form of communication is followed by a brief explanation:

- **Point-to-Point** - The Point-to-Point function of TCP/IP serves three main components: a method for encapsulating datagrams over serial lines; an extensible Link Control Protocol to configure and test data-link connections; and a method for establishing different network-layer protocols or Network Control Protocols (NCP). They include CTC and IUCV connections.
- **Ethernet Network** - A local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection.

Server Functions

Following are the server functions and a description of each. These functions may be used to provide shared services to workstations over a network.

- **DNS** - The DNS (Domain Name System), commonly referred to as simply a *name server*, maps a host name to an internet address or an internet address to a host name.
- **FTP** - The File Transfer Protocol virtual machine (FTPSERVE) serves client requests from the TCPIP virtual machine through VMCF communication. The FTP server allows you to transfer files between your

local host and a foreign host that supports TCP/IP. When invoked, FTP establishes a connection to a foreign hosts FTP server. After you have identified yourself to the foreign FTP server, you can retrieve information about the foreign server, list files, transfer files, delete files, rename files and execute CMS commands.

- **LDAP** - The LDAP server (LDAPSRV) provides directory services for clients according to the Lightweight Directory Access Protocol. The server interoperates with any LDAP 2 or LDAP 3 directory client. A directory is like a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. As a consequence, directories do not usually implement the complicated transaction or rollback schemes that relational databases use for doing high-volume complex updates.
- **MRoute** - The MRoute server provides an alternative to the static TCP/IP gateway definitions. For IPv4, MRoute implements the OSPF protocol described in RFC 1583 (OSPF Version 2) and the RIP protocols described in RFC 1058 (RIP Version 1) and in RFC 1723 (RIP Version 2). For IPv6, MRoute implements the IPv6 OSPF protocol described in RFC 2740 (OSPF for IPv6) and the IPv6 RIP protocol described in RFC 2080 (RIPng for IPv6).

Note: If you previously used the Routed server to provide dynamic routing services for your installation, you need to migrate to the MRoute server for these services.

- **NFS** - The Network File System virtual machine (VMNFS) allows a client system to access CMS minidisks, SFS directories, and BFS directories as if they were local to the client. VMNFS allows users to execute programs, and to create and edit files. The VMNFS virtual machine requires access to the IBM Language Environment® runtime library during execution.
- **Port Mapper** - The Port Mapper (PORTMAP) server application is used to map program numbers and various numbers to RPC programs that request information. The PORTMAP server only knows about RPC programs on the local host system. The calling application contacts PORTMAP on the destination host to obtain the correct port number of a specific remote program.
- **RExec** - The Remote Execution virtual machine (REXEC) sends a single command to a remote system for execution. The remote system name, user ID, password and command string can all be passed as parameters. The REXEC server can log on, execute the command and log off. REXEC can also be used for network management purposes. Procedures to start and stop TCP/IP devices (physical links) can be initialized using the REXEC command and the OBEYFILE command.
- **SMTP** - The Simple Mail Transfer Protocol virtual machine (SMTP) operates as a mail gateway between TCP/IP network sites and RSCS network sites. This means, for example, OfficeVision® users can exchange mail with UNIX workstations through the VM TCP/IP SMTP gateway. The SMTP server allows you to create custom mail headers (RFC822), provides an interface which allows the querying of SMTP mail delivery queues and provides a set of privileged commands for system administrator tasks. SMTP can be configured to use either a domain name server or the local site tables.
- **SNMP** - SNMP is an architecture that allows you to manage an internet environment. You can use SNMP to manage elements such as gateways, routers, and hosts on the network.
- **SSL** - The Secure Socket Layer (SSL) server, which runs in one of several "pool" virtual machines (SSLnnnnn), provides processing support for secure (encrypted) communication between remote clients and z/VM TCP/IP application servers that are configured for secure communications. The TCP/IP (stack) server routes requests for secure connections to an SSL server, which interacts with a client on behalf of an application server to perform handshake operations and the exchange of cryptographic parameters for a secure session. The SSL server then manages the encryption and decryption of data for an established, secure session. The SSL server performs similar functions for selected z/VM client commands that incorporate support for secure communications.
- **TELNET** - The Teletypewriter Network (TELNET) feature is part of the TCPIP virtual machine. The TELNET protocol provides a standardized interface, through which a program on one host (the TELNET client) may access the resources of another host (the TELNET server) as though the client were a local terminal connected to the server. TELNET provides interactive access to local and remote hosts, support for TN3270 and TN3270E, but does not support graphics.
- **UFTD** - The UFT daemon (UFTD server) accepts unsolicited file transfer requests to receive files from remote clients for local users.

Client Functions

- **FTP** - The FTP client sends requests to the FTP server virtual machine.
- **LDAP** - z/VM supports the following LDAP client utilities: LDAPADD (ldapadd), LDAPCHPW (ldapchangepwd), LDAPCMPR (ldapcompare), LDAPDLET (ldapdelete), LDAPMDFY (ldapmodify), LDAPMRDN (ldapmodrdn), and LDAPSrch (ldapsearch). z/VM supports the following backend utilities: DS2LDIF (ds2ldif), DB2PWden (db2pwdn), and LdapEXOP (ldapexop).
- **LPR** - the LPR command is used to communicate with an LPD server to submit commands and documents to be printed.
- **NFS** - The NFS client allows a remote file system to be logically included as a component of a Byte File System.
- **NOTE** - The NOTE command (provided with CMS) allows users to send mail throughout the local network and to external network sites.
- **NSLOOKUP** - The NSLOOKUP command is used to query names and allows you to locate information about network nodes, examine the content of a name server database and establish the accessibility of name servers.
- **REXEC** - The REXEC client in VM/CMS issues requests to execute commands on other network systems.
- **SENDFILE** - The SENDFILE command (provided with CMS) allows users to send files throughout the local network and to external network sites.
- **TELNET** - The TELNET client allows you to log on from any VM/CMS terminal to either local or remote networks that are operating TCP/IP.

Network Status and Management Functions

- **NETSTAT** - The NETSTAT command displays the network status of the local host, TCP/IP connections, network clients, routers, devices and the TELNET server. NETSTAT will provide information on active TCP connections, all TCP/IP servers on the local host, devices, links and IP routing tables.
- **OBEYFILE** - The OBEYFILE command allows you to execute the TCP/IP configuration statements while TCP/IP is running, thus allowing updates to occur without halting TCP/IP operations. Primarily, OBEYFILE is used to change the TCP/IP system temporarily while TCP/IP is running.
- **PING** - The PING command tests the connection to any TCP/IP host.
- **MPRoute** - The Multiple Protocol Route (MPRoute) virtual machine implements the OSPF, IPv6 OSPF, RIP and IPv6 RIP protocols, providing an alternative to the use of static TCP/IP gateway definitions. When properly configured, the VM host running with MPRoute becomes an active OSPF or RIP (or both) router in a TCP/IP network.

The Open Shortest Path First protocol (OSPF) is an interior gateway protocol. It distributes routing information between routers that belong to a single autonomous system; that is, a group of routers that all use a common routing protocol.

- **SNMP** - The Simple Network Management Protocol (SNMP) virtual machine allows you to manage your TCP/IP network. SNMP is an internet standard that defines a set of functions that may be used to monitor and control network elements. The SNMP server (or agent) responds to commands issued by the client (or monitor).

Application Programming Interfaces

- **RPC** - The Remote Procedure Call is an application programming interface (API) for developing distributed applications. It allows programs to call subroutines that are executed at a remote system. The caller program (client) sends a call message to the server, and waits for a reply message. In order to send a message the caller must know the exact port number used by the RPC program. To get this port number, the caller sends a request to the PORTMAP server on the remote host to retrieve the port information for the desired program.
- **SNMP DPI** - The Simple Network Management Protocol Daemon (SNMPD) is a server that communicates management information between network management stations and agents in the

network. The **SNMP Distributed Programming Interface (DPI)** allows users to manipulate the information or MIB variables for each layer in the TCP/IP protocol.

- **Sockets** - The Sockets API provides a simplified procedure to transfer information and to communicate between applications. The socket interface is designed to provide applications with a network interface that hides the details of the physical network.

Migration Information and Resources

Information about TCP/IP changes and enhancements that can (or will) affect migration to the current level of TCP/IP for z/VM can be obtained through the following resources:

- *Program Directory for TCP/IP for z/VM.*
- TCP/IP for z/VM (<https://www.ibm.com/vm/related/tcpip>).

Note: TCP/IP is supported on associated releases of CP and CMS only.

User ID Privilege Class Considerations

To facilitate various TCP/IP administrative functions and allow for certain capabilities to be used by various TCP/IP servers, specific privilege classes (other than Class G) have been defined for some TCP/IP user IDs within the IBM-supplied z/VM (CP) user directory. These privilege classes have been assigned to these user IDs for the purposes stated in Table 2 on page 4:

Table 2. TCP/IP Server and User ID Assigned Privilege Classes

User ID	Privilege Class	Pertinent Commands and Capabilities
FTPSERVE	D	CP QUERY RDR / CP PURGE RDR command capability, for manipulating files for a specific user ID. Required when FTP virtual reader support is enabled.
REXECD	B	CP XAUTOLOG capability, to logon a specific user ID or rexec agent machine.
SSLDCSSM	E	CP DEFSEG / CP SAVESEG command capability, to allow for definition of a Discontiguous Saved System (DCSS).
SSLnnnnn, SSLSERV	C	CP FOR command capability, to accommodate coordinated virtual machine dump processing
SNMPSUBA	E	Diagnose X'26C' capability, to allow device MAC addresses to be obtained.
TCPMAINT	A	FORCE. Used to force a logoff of an inoperative TCP/IP server virtual machine.
	B	XAUTOLOG. Used to allow for the manual startup of an inactive TCP/IP server virtual machine.
	C	General system programmer-related functions. Specific commands for which this class has been assigned are: TRSAVE/TRSOURCE. Used to allow for the activation of TCP packet traces and the management of resulting trace data files.

Table 2. TCP/IP Server and User ID Assigned Privilege Classes (continued)

User ID	Privilege Class	Pertinent Commands and Capabilities
TCPIP	A	FORCE. Used to force a logoff of an inoperative TCP/IP server virtual machine.
	B	XAUTOLOG. Used to allow for the manual startup of an inactive TCP/IP server virtual machine. ATTACH/ DETACH, VARY, ENABLE/DISABLE SNA, DEFINE/MODIFY/DELETE SWITCH/LAN. Used to manage the devices and interfaces that provide TCP/IP connectivity.
Various	B	DMSLINK CSL capability, to verify or obtain access to minidisks on behalf of a user. Servers that require this privilege class include: FTP (FTPSERVE, CSMERVE), NFS (VMNFS), REXEC (REXECD), SMTP (SMTP).
Various	B	MSGNOH command capability; servers that require this privilege class include: FTP (FTPSERVE, CSMERVE), LDAP (LDAPSRV), MPRoute (MPROUTE), SMTP (SMTP), UFT (UFTD).

For the TCPIP user ID, the indicated privilege classes are also required to allow the use of certain CP DIAGNOSE commands that facilitate server operations.

Note: All user IDs listed in the TCP/IP OBEY list (that is, users who are authorized to issue privileged TCP/IP functions) have access to **all** CP commands that are available to the TCPIP user ID.

If necessary, modify the privilege classes associated with these user IDs to comply with any policies or guidelines established for your installation about such privileges. Doing so might require coordination with any CP privilege class modification that might have been implemented by your installation.

Alternatively, it might be appropriate to use the CP MODIFY command to adjust the privilege classes for certain CP commands (for example, the CP SHUTDOWN command, which has an IBM-defined privilege class of A) so that a separate, site-defined privilege class is required for their use.

User ID Minidisk Considerations

Several TCP/IP servers require the use of a work disk to which files can be written to accomplish a given task. These tasks might involve the creation of files for server initialization, debugging or activity logging, or handling error notifications. For certain TCP/IP servers, the server 191 disk serves as a designated work disk, and therefore must be accessed Read/Write.

TCP/IP servers to which this requirement applies are the:

- TCP/IP (Stack) server
- FTP server
- NFS server
- SMTP server
- SSL server (when not defined as a pool server)

If the 191 disk for the listed servers is not available Read/Write, server operation might be inhibited, and some task functions might not be available.

Shared File System (SFS) Considerations

In general, an SFS directory cannot be substituted in place of a 191 minidisk for any of the TCP/IP servers supplied with TCP/IP for z/VM.

However, when an SSL server pool is implemented, individual 191 minidisks for these servers cannot be used. For such servers, an SFS root directory must be defined, in addition to a common-use "work space" directory, to which all pool servers have Read/Write access. The SSLPOOL command can be used (as needed) to create these SFS directories and establish the appropriate authorizations for an SSL server pool. For more information, see ["SSLPOOL Command"](#) on page 502.

Implications of Assigning Different Server Virtual Machine Names

In general, the amount of customization required to install TCP/IP or to configure a protocol server can be minimized by using IBM-supplied, default virtual machine names (user IDs).

If non-default server machine names must be used in your environment (for example, to conform to local site naming standards), this might be accommodated by duplicating existing servers, and using an updated SYSTEM DTCPARMS file in conjunction with a global server exit that provides common server processing.

However, for some environments it may be necessary to modify various TCP/IP configuration files and product executables to accommodate specific user ID changes. The information that follows is provided to assist you with making changes of this nature.

Accommodating Changed Server Names

If you elect to change the server user ID (or, name) for any of the TCP/IP virtual machines that provide services in your environment, certain modifications must be made. As you use the configuration steps provided later in this publication, ensure the following changes are made to account for server naming differences:

1. Create a SYSTEM DTCPARMS file that identifies and further defines each TCP/IP protocol server that is used in your environment. Model your server definitions after those in the definition section of the supplied IBM DTCPARMS file.
2. Update the TCP/IP server configuration file (PROFILE TCPIP, or its equivalent) to reflect any server name changes. Statements within this file that are likely to be affected by name-related changes are:
 - AUTOLOG
 - OBEY
 - PORT
 - SSLSERVERID
3. When the user ID of the TCP/IP server is changed from the default of **TCPIP**, one or more of the following changes will be required, depending on your environment:
 - Modify the **TCPIPUSERID** statement within the TCPIP DATA file to reflect the correct TCP/IP server virtual machine user ID.

Example:

```
TCPIPUSERID TCIPA1
```

- For the RSCS UFT client or RSCS Print Server, include the **TCPIP=** operand as part of any **PARM** statements that correspond to LPD or UFT link definitions. This operand allows you to identify the TCP/IP server machine that is in use.

Example:

```
...  
LINKDEFINE LPD TYPE LPD  
PARM LPD EXIT=LPDXMANY TCPIP=TCIPA1  
...
```

Configuration files that may require such additions are:

- RSCSLPD CONFIG

- RSCSLPR CONFIG
 - RSCSLPRP CONFIG
 - RSCSTCP CONFIG
 - RSCSUFT CONFIG
4. When the user ID of the Simple Mail Transfer Protocol (SMTP) server is changed from the default of **SMTP**, one or more of the following changes will be required, depending on your environment:
- Modify the **SMTPSERVERID** statement within the TCPIP DATA file to reflect the correct SMTP server virtual machine user ID.

Example:

```
SMTPSERVERID SMTPA1
```

5. If RACF® is active on the system on which TCP/IP is installed, and:
- you elect to use a different name for the File Transfer Protocol (FTP) server or the Network File System (NFS) server, or
 - you elect to use multiple FTP or NFS servers,
- any modified or additional server names must be identified (or made available for reference) whenever the FTPPERM or NFSPERM execs are used to establish authorization for these servers to act on a users' behalf. For more information see [Appendix A, “Using TCP/IP with an External Security Manager,”](#) on page 685.
6. By default, the **TCPMaint** user ID is designated as the **owner** of the various TCP/IP resources that are placed into production in your environment. As such, this user ID often serves as the focal point for administering TCP/IP protocol servers and services. When a different user ID is used as the TCP/IP resource **owner** or **administrator**, one or more of the following changes may be required, depending on your environment:
- Update the TCP/IP server configuration file (PROFILE TCPIP, or its equivalent) to account for the changed TCP/IP administrative user ID. Statements within this file that are likely to be affected by such changes are:
 - INFORM
 - OBEY
 - Update the SYSTEM DTCPARMS file to include appropriate **:Owner** tags to identify your administrative user ID for each protocol server used in your environment.

Example:

```
:owner.TCPADMIN
```

- For the SMTP server, the TCPMAINT user ID is assumed as the effective user ID for the statements that follow (when these statements are omitted from the SMTP server configuration file, SMTP CONFIG):
 - BADSPOOLFILEID
 - POSTMASTER

In addition, the supplied sample SMTP configuration file includes these statements, for which the TCPMAINT user ID has been specified:

- ONDISKFULL
- SMSGAUTHLIST

Customize these statements, as required, to reflect the administrative user ID that is in use.

- For the NFS server, update the VMNFSCMS exit exec (if used) to identify the correct administrative user ID.

Example:

```
...  
if origin = "TCPADMIN" then signal good  
...
```

- For the SSL server pool, update the SYSTEM DTCPARMS file to include an SSL server entry and :Admin_ID_list. definition that cites the appropriate administrative user IDs:

Example:

```
...  
:nick.SSL* :type.server :class.ssl  
:Admin_ID_list.TCPADMIN GSKADMIN  
...
```

7. By default, the GSKADMIN user ID is designated as the owner of the BFS filespace in which the SSL server key database resides, and as the owner of the filespace used as SSL server work space. The GSKADMIN user ID is also defined as an SSL server administrative user, and as administrator of the key database. When a different user ID is used for these purposes, one or more of the following changes may be required, depending on your environment.

- For the SSL server pool, update the SYSTEM DTCPARMS file to include an SSL server entry and :Admin_ID_list. definition that cites the appropriate administrative user IDs:

Example:

```
...  
:nick.SSL* :type.server :class.ssl  
:Admin_ID_list.TCPMAINT SSLADMIN  
...
```

- Update the GSKSSLDB LOADBFS file such that the appropriate user ID is designated as the GSKSSLDB file space owner.

Example:

```
...  
DEFAULT_OWNER          ssladmin  
...
```

- Update the SSLSERV LOADBFS file such that the appropriate user ID is designated as the SSLSERV file space owner.

Example:

```
...  
DEFAULT_OWNER          ssladmin  
...
```

Note: Make all changes to the GSKSSLDB and SSLSERV LOADBFS files through use of a VMSES/E local modification. The updated LOADBFS file then must be reprocessed using the LOADBFS command by an appropriate file pool administrative user ID, such as the TCP/IP for z/VM installation and service user ID (5VMTCP40).

8. When the SNMP Query Engine server name is changed, the **SNMPQE** statement within the SNMPARMS NCCFLST file must be updated to reflect this change.

Example:

```
...  
SNMPQE  SNMPQE1          * Userid of the SNMP Query Engine  
...
```

9. When the LDAP server name is changed:

- Update the LDAPSrv LOADBFS file so that file space allocation and ownership are correct for the renamed server.

Example:

```

...
SETVAR fspace_ldap_ LDAPSRVRB
SETVAR own_id_ldap_ LDAPSRVRB
...

```

Note: Make all changes to the LDAPSRV LOADBFS file through use of a VMSES/E local modification. The updated LOADBFS file then must be reprocessed using the LOADBFS command by an appropriate file pool administrative user ID, such as the TCP/IP for z/VM installation and service user ID (5VMTCP40)

Multiple Server Instance Restrictions

For a given VM TCP/IP host, multiple TCP/IP server machines can be defined and used for nearly all protocol server classes, as required. However, there are several servers for which the use of multiple server instances is not supported (not recommended). Servers to which this restriction applies (that is, for which only one instance should be employed for a given TCP/IP stack server) are the:

- MPRoute server (MPROUTE)
- SSL server — when not defined as a pool server.

The restriction against multiple instances for these servers stems from the nature of the protocol service that each provides.

Note:

- The TCP/IP stack server might not explicitly prevent using multiple instances of the listed servers. However, be aware that the result of doing so is unpredictable.
- To employ multiple SSL servers, an SSL server pool configuration should be used. For more information, see [Chapter 15, “Configuring the SSL Server,” on page 453](#).

Mutually Exclusive Servers

As improvements and additional TCP/IP support functions are incorporated within TCP/IP for z/VM, additional servers might be introduced that provide services similar to those provided by an existing server. In some cases, the services offered by a newer server may surpass or otherwise overlap those provided by an existing server. When this occurs, certain TCP/IP protocol servers may then become mutually exclusive with respect to use.

When you install and configure TCP/IP for z/VM, determine which of these protocol servers best addresses the requirements of your environment, and then employ only that server.

Publication References

Additional TCP/IP, z/VM, and VMSES/E product information can be obtained in the following publications:

- *z/VM: TCP/IP Programmer's Reference*
- *z/VM: TCP/IP User's Guide*
- *z/VM: TCP/IP Messages and Codes*
- *z/VM: TCP/IP Diagnosis Guide*

Chapter 2. System Requirements for TCP/IP

This chapter identifies the system requirements for TCP/IP for z/VM.

z/VM Device Definition Considerations

Most network devices do not require definition in the system configuration file or HCPRIO. Device attributes are determined dynamically through the device initialization process. For more information on when and how devices are defined for z/VM, see [z/VM: CP Planning and Administration](#).

Hardware Environment

TCP/IP for z/VM operates on any IBM or z/Architecture® processor supported by z/VM (or any compatible processor). For information about supported processors, see [z/VM: General Information](#).

TCP/IP also requires a 3270-equivalent workstation for TCP/IP administration.

Network Attachments

TCP/IP for z/VM requires a network processor and associated components for attachments to the teleprocessing network. The following are possible network attachments that you may have installed or plan to install that work in conjunction with TCP/IP or a group of OSA-Express adapters. For information about supported network attachments, see [z/VM: General Information](#).

Open System Adapter-Express (OSA-Express)

TCP/IP for z/VM provides support for the OSA-Express adapter. This adapter provides fully-integrated native-systems connectivity to a local area network (LAN) from an IBM Z processor. With the OSA-Express adapter, a type of I/O called the Queued Direct I/O Hardware Facility (QDIO) is available. QDIO allows the TCP/IP stack to exchange data directly with an I/O device, without performing traditional I/O instructions. Data transfer is initiated and performed by referencing main storage directly through a set of data queues by the I/O device and the TCP/IP stack. After the TCP/IP stack establishes and activates the data queues, there is minimal processor intervention required to perform the direct exchange of data.

Multiple guests can exchange data directly with a single OSA-Express adapter or a group of OSA-Express adapters using QDIO when they couple to a virtual switch using a virtual QDIO NIC.

The OSA-Express 1000Base-T models can also be configured in non-QDIO mode (CHPID type OSE) for use in LAN Emulation mode using the LAN Channel Station (LCS) protocol.

HiperSockets

As an extension to the Queued Direct I/O Hardware Facility, HiperSockets provides a way for programs to communicate within the same logical partition (LPAR) or to any logical partition within the same Central Electronics Complex (CEC) using traditional TCP/IP socket connections.

Channel-to-Channel Support

Channel-to-Channel connections are supported using the IBM 3088 Multi-system Channel Communication Unit. TCP/IP supports direct connection to another VM TCP/IP, MVS, z/OS®, or Linux® on IBM Z® using the IBM 3088.

TCP/IP also supports direct connections to other systems using IBM ESCON and FICON® Channel-to-Channel Adapters.

IUCV

The CP IUCV Service may be used to connect TCP/IP service machines on the same VM host. IUCV may also be used to connect VM TCP/IP to Linux on IBM Z.

z/VM Virtual Network Adapters

TCP/IP provides support for OSA-Express and HiperSockets virtual network adapter types. These adapters are fully-simulated devices that appear as physical adapters to the TCP/IP stack. Support is provided for these virtual network adapters (network interface cards or NICs) through the existing TCP/IP DEVICE and LINK configuration statements for these hardware types. Virtual network adapters are created by the DEFINE NIC command in CP and are used to connect a virtual machine (for example, TCP/IP) to a virtual LAN (guest LAN) or a virtual switch. For more information on z/VM's virtual networking support, see [z/VM: Connectivity](#).

Software Environment

TCP/IP function level 730 requires:

- z/VM 7.3 (TCP/IP function level 730 cannot be used with previous levels of CP and CMS)
- Language Environment supplied with z/VM 7.3 (previous levels of Language Environment cannot be used.)

To develop programs in Pascal or to modify TCP/IP Pascal components, use IBM VS Pascal 1.2 Compiler and Library (5668-767).

Note: The Pascal run-time library is included with TCP/IP.

To develop applications in the C programming language, or to modify TCP/IP C components, the following is required:

- IBM C for VM/ESA Compiler 3.1 (5654-033)

Language Environment has been used to build the C components that provide the following TCP/IP services:

- LDAP Server and Client
- MPRoute Server
- NFS Client and Servers
- PortMapper Server
- REXEC Daemon
- SNMP Agent
- SNMP Query Engine
- Sockets Applications Programming Interface

TCP/IP function level 730 exploits a subset of RSCS function that is available with z/VM 7.3. This subset provides enhanced printing support through RSCS LPR and LPD functions, along with TN3270E protocol support for printer sessions. This is the only intended or implied use of this subset of RSCS. Customers requiring the use of other RSCS functions must have or acquire a license for the RSCS Networking for z/VM feature.

If you are running a primary or secondary name server, you need Db2® Server for VM Version 6 (5648-A70) or Db2 Version 7 (5697-F42). You do not need Db2 Server for VSE & VM if you are running a caching-only name server.

To link two TCP/IP virtual machines using the VM/Pass-Through Facility (PVM):

- The connections may be linked using the PVM IUCF peer-to-peer connection facility. The PVM virtual machines on the nodes running TCP/IP must be at PVM Version 2 (5684-100).

Chapter 3. Defining the TCP/IP System Parameters

This chapter describes how to define the system parameters that are required by the TCP/IP client programs.

Configuring the TCPIP DATA File

The TCPIP DATA file defines system parameters used by TCP/IP client applications. It is used to specify configuration information for single or multiple host systems. For example, if you have multiple versions of TCP/IP installed with a unique set of server virtual machine names for each version, you can specify which TCP/IP product to use in this file. It also allows you to specify:

- Host name of the VM host
- User ID of the TCPIP virtual machine
- Domain origin of the host
- Output trace
- Name server specifications
 - Name server addresses and default loopback address
 - Foreign port of the name server
 - Number of seconds resolver should wait while trying to communicate with the name server
 - Number of seconds between connection attempts
 - Communication method between resolver and name server

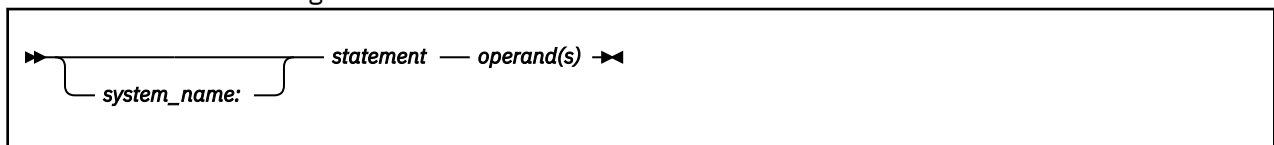
A sample TCP/IP DATA file is shipped as TCPIP SDATA on the TCPMAINT 592 disk. It should be copied to TCPIP DATA (on the same disk) and customized.

Change the configuration statements in the TCPIP DATA file to define the client parameters. You can specify configuration information for single or multiple systems in a single file, the TCPIP DATA file.

Statement Syntax

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

The format for each configuration statement is:



Where:

system_name

is an optional 1- to 8-character qualifier that can be specified before a configuration statement. The use of such a qualifier allows configuration information for multiple systems to be specified in a single TCPIP DATA file.

A statement that is qualified in this manner is in effect only if *system_name* matches the name of the system, as determined by the CMS IDENTIFY command. Statements that are not matched are ignored. A statement that is not qualified by a *system_name* is in effect on every system.

statement

is a TCPIP DATA file configuration statement.

operand(s)

is one or more configuration statement operands.

This section describes the syntax of the configuration statements for the TCPIP DATA file.

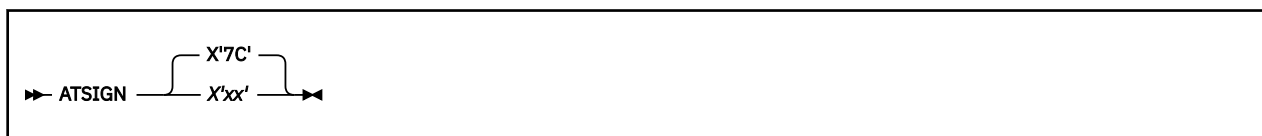
ATSIGN statement

Purpose

The ATSIGN statement specifies an alternate hexadecimal value to be used for the "at sign" (@) character. An alternate definition for this character is required only when your site is using code pages in which the EBCDIC @ character is not defined as X'7C'. If this statement is not specified, only the X'7C' value will be in effect.

The alternate definition you provide is *only* used by the SMTP server and by certain CMS commands (such as SENDFILE) that process files directed to this server. The alternate value must match the @ EBCDIC codepoint defined by the national code page in use.

The supplied definition is used (in addition to the X'7C' default) to correctly map the EBCDIC @ value to its ASCII equivalent (X'40') when internet address information is processed. This ensures that internet addresses are properly handled for mail that is inbound to, or outbound from, your site.



Operands

X'xx'

Specifies the hexadecimal value used to define the @ character; xx is a hexadecimal character specification.

For example, in Finnish/Swedish code page 278, the @ character is defined as the hexadecimal X'EC'. When this codepage is in use the ATSIGN statement should be specified as:

```
ATSIGN X'EC'
```

Additionally, the appropriate translate table (for example, 02780819 TCPXLBIN) must be available for use by the SMTP virtual machine, as file SMTP TCPXLBIN.

Usage Notes

- Modifications to the @ character through the TCPIP DATA file ATSIGN statement cannot be used by installations that make use of RFC 822 header rewrite rules. The processing of rules defined in the SMTP RULES file is table driven; the logic to support @ character redefinition within these tables has not been incorporated in this release of TCP/IP.
- The ATSIGN statement should not be used to account for differences or problems that arise with the symbol used for the virtual console delete character function (the system default symbol for this function is the @ character). The CP TERMINAL CHARDEL command should instead be used to resolve character deletion problems when the @ character is used.

DOMAINLOOKUP statement

Purpose

The DOMAINLOOKUP statement specifies what methods are used to resolve host names and in what order these methods are used. By default, name resolution is attempted by asking a question of each name server and then checking for the answer in the local host files (using the ETC HOSTS file if it is

present, or using the site tables created by the HOSTS LOCAL file if ETC HOSTS is not present). The resolution is complete when the first answer is found. You can use the DOMAINLOOKUP statement to instruct the resolver to use *only* name servers to resolve host names, or to attempt to find an answer in the local files before any name servers are contacted.

Note: IPv6 addresses are preferred over IPv4 addresses.



Operands

DNS

Use both the name servers specified by the NSINTERADDR statements and the local site table to perform host name resolution. Name servers are used before the local site table.

FILES

Use both the local site table and the name servers specified by the NSINTERADDR statements to perform host name resolution. The content of the local site table is used before any name servers.

DNSONLY

Use only the name servers specified by the NSINTERADDR statements to perform host name resolution.

FILESONLY

Use only the local site table to perform host name resolution.

Note: For options in which local files are used and an ETC HOSTS file exists, HOSTS LOCAL file information is not used.

Usage Notes

- When DOMAINLOOKUP DNS is in use and the resolver is required to ask more than one question to obtain an answer, each question is asked of each name server in turn and the ETC HOSTS file, if present, is consulted last; or, if the ETC HOSTS file is not present, the HOSTS SITEINFO file is consulted last. In other words, the local host files are consulted after the first question has been asked of all name servers.

For more about name resolution, see [“DOMAINSEARCH statement ” on page 16.](#)

DOMAINORIGIN statement

Purpose

The DOMAINORIGIN statement specifies the domain that is appended to the host name in the HOSTNAME statement to form the fully qualified domain name (FQDN) for a system.

The domain provided on the DOMAINORIGIN statement is also used as a DOMAINSEARCH value for the purposes of host name resolution. The rules that govern how host name resolution is performed are described in the [“DOMAINSEARCH statement ” on page 16.](#)

➤➤ DOMAINORIGIN — *domain* ➤➤

Operands

domain

Specifies the domain origin that is appended to the host name when the complete local host name is formed. The *domain* value is also used as a domain for host name resolution, as if it were specified in a DOMAINSEARCH statement.

The value for *domain* must conform to the following rules:

- Must contain one or more tokens separated by a period.
- Each token must be at least one character and a maximum of 63 characters.
- Each token must start with a letter or number.
- Remaining characters in each token must be a letter, number, or hyphen.
- The length of the host name plus the length of the domain name must be less than or equal to 255.

Examples

This example appends the domain origin of SYS1.DOMAIN.NAME to the host name:

```
DOMAINORIGIN SYS1.DOMAIN.NAME
```

Usage Notes

- Case translation is not performed on the domain origin.
- The DOMAINORIGIN configuration statement must be customized at each site.

DOMAINSEARCH statement

Purpose

The DOMAINSEARCH statement specifies a domain to be used when host names are resolved. Up to 50 DOMAINSEARCH statements can be used. Case translation is not performed on the domain name provided. Currently, name resolution throughout the internet is case independent.

The order of the statements encountered in the TCPIP DATA file is the order they will be used during host name resolution. Any DOMAINSEARCH configuration statements must be customized at each site. Note that the domain specified for the DOMAINORIGIN statement is also used for host name resolution, as if it appeared in a DOMAINSEARCH statement.

```
➡ DOMAINSEARCH — domain ➡
```

Operands

domain

Specifies a domain to be added to the list of domains to be used when host name resolution is performed.

Host name resolution is a process that attempts to obtain an IP address for a given host. By convention, host names are considered to be composed of two parts — the *host* portion and a *domain* portion.

It is common for a local environment to make use of more than one domain name. For example, the **misery.movie.edu** domain and the **comedy.movie.edu** domain might both be in use. Assume that when you want to resolve the host **tootsie**, you would like both domains to be searched when name resolution is performed. This can be accomplished by specifying each domain on a DOMAINSEARCH statement. Note that if one of these domains is already specified on the DOMAINORIGIN statement, a DOMAINSEARCH statement for that domain is not required.

The resolver will actually ask multiple questions of the specified name servers until it gets an answer, at which time it ends the resolution process. For example, if the host **tootsie** exists in more than one of the domains defined for the DOMAINORIGIN and DOMAINSEARCH statements, the host address returned will be that for the domain defined first in the TCPIP DATA file. To avoid the chance of getting an address from a domain that is not desired, a fully-qualified host name, such as **tootsie.comedy.movie.edu**, should be specified when TCP/IP services are used.

The nature of the name you attempt to resolve can also affect the way the search is done. If the name you provide is followed by a period (.), that name (with the trailing period removed) is assumed to be fully qualified already, and the domains from the DOMAINSEARCH and DOMAINORIGIN statements are not used. If the name contains (but does not end with) a period, it is presumed this name may already be fully qualified; the name, as provided, is used for the search before trying any names created by appending the domains from the DOMAINSEARCH and DOMAINORIGIN statements. If the name does not contain a period, it is presumed to not be fully qualified, so names created by appending the domains from the DOMAINSEARCH and DOMAINORIGIN statements are tried. If none of these names result in an answer, the name, as originally specified, is tried as a final attempt.

Examples

The examples that follow attempt to illustrate how the input name and the content of the TCPIP DATA file are used to resolve a name. Assume the TCPIP DATA file contains these definitions:

```
DomainSearch      tragedy.movie.edu
DomainOrigin      comedy.movie.edu
DomainSearch      suspense.movie.edu
DomainSearch      movie.edu
```

If one issues this PING command:

```
ping tootsie
```

these questions are sent to the name server:

```
tootsie.tragedy.movie.edu
tootsie.comedy.movie.edu   (this question is answered)
```

For this command:

```
ping tootsie.comedy
```

these questions are sent to the name server:

```
tootsie.comedy
tootsie.comedy.tragedy.movie.edu
tootsie.comedy.comedy.movie.edu
tootsie.comedy.suspense.movie.edu
tootsie.comedy.movie.edu   (this question is answered)
```

If this command is issued:

```
ping tootsie.comedy.
```

the questions sent to the name server are:

```
tootsie.comedy
(the search fails)
```

Finally, if this PING is performed:

```
ping junk
```

these questions are sent to the name server:

```
junk.tragedy.movie.edu
junk.comedy.movie.edu
junk.suspense.movie.edu
```

```
junk.movie.edu
junk
(the search fails)
```

HOSTNAME statement

Purpose

The HOSTNAME statement specifies the TCP host name of this VM host. The fully qualified domain name for the host is formed by concatenating this host name with the domain origin (specified by the DOMAINORIGIN configuration statement). If the host name is not specified, the default host name is the node name returned by the CMS IDENTIFY command.

➤ HOSTNAME — *hostname* ➤

Operands

hostname

Specifies the TCP host name of the VM host.

Because the hostname is used to build the fully qualified domain name (FQDN), the value specified must conform to the following domain name rules:

- Maximum of 63 characters.
- Must contain one or more tokens separated by a period.
- Each token must be at least one character and less than 64 characters.
- Each token must start with a letter or number.
- Remaining characters in each token must be a letter, number, or hyphen.

Examples

The TCPIP DATA file is shared between two systems, ZVMSYS1 and ZVMSYS4. The HOSTNAME statements define the host name on each system:

```
ZVMSYS1: HOSTNAME SYS1TCPIP
ZVMSYS4: HOSTNAME SYS4TCPIP
```

Usage Notes

Case translation is not performed on the host name.

HOSTVERIFICATION statement

Purpose

The HOSTVERIFICATION statement provides the default client host verification setting when the SECURE option is specified on the TELNET command, but no host verification option (HVCONTINUE, HVNONE, or HVREQUIRED) is specified. Specifying a host verification option on the TELNET command overrides this option.

➤ { HOSTVERIFICATION CONTINUE
HOSTVERIFICATION NONE
HOSTVERIFICATION REQUIRED } ➤

Operands

HOSTVERIFICATION CONTINUE

Verifies that the host name, domain name, or IP address in the server certificate matches what was specified on the TELNET command. If they do not match, the handshake is allowed to continue.

HOSTVERIFICATION NONE

Specifies that no host verification will be performed.

HOSTVERIFICATION REQUIRED

Verifies that the host name, domain name, or IP address in the server certificate matches what was specified on the TELNET command. If they do not match, the handshake will fail.

NSINTERADDR statement

Purpose

The NSINTERADDR statement defines the internet address of a name server. This parameter can be repeated, without limit, to specify the internet addresses of alternative name servers. Connections to the name servers are attempted in the order they appear in this configuration file. If NSINTERADDR statement is not coded in the TCPIP DATA file, the resolver does not attempt to use a name server and looks up all domain names in the local site tables. For more information on the local site tables, see Chapter 4, “Configuring the Local Host Files,” on page 27.

```
➤ NSINTERADDR — internet_address ➤
```

Operands

internet_address

Specifies the internet address of a name server.

Usage Notes

1. You must code the values for a name server IPv4 address in dotted decimal format. The following restrictions apply:
 - You cannot specify the following IPv4 addresses as valid name server IPv4 addresses:
 - IPv4 unspecified address (0.0.0.0)
 - IPv4 broadcast address
 - IPv4 multicast address
 - You cannot specify IPv4 subnet length information as part of the IPv4 address.
2. You must code the values for a name server IPv6 address in colon hexadecimal format. You can specify the IPv6 addresses in upper case, lower case, or mixed case formats. The following restrictions apply:
 - You cannot specify the following IPv6 addresses as valid name server IPv6 addresses:
 - IPv6 unspecified address (::)
 - IPv6 multicast address
 - IPv4-mapped address in dot-decimal format
 - IPv4-compatible IPv6 address
 - You cannot specify scope information as part of the IPv6 address.
 - You cannot specify prefix length information as part of the IPv6 address.

NSPORTADDR statement

Purpose

The NSPORTADDR statement specifies the port of the name server.



Operands

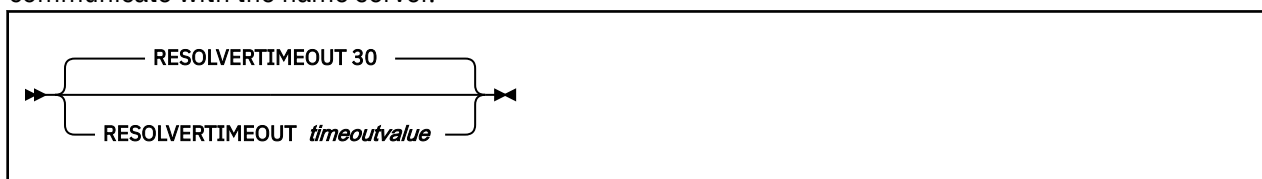
nsportaddr

Specifies the port of the name server. The range is 1 through 65 535. The default is port 53.

RESOLVERTIMEOUT statement

Purpose

The RESOLVERTIMEOUT statement specifies the number of seconds the resolver waits while trying to communicate with the name server.



Operands

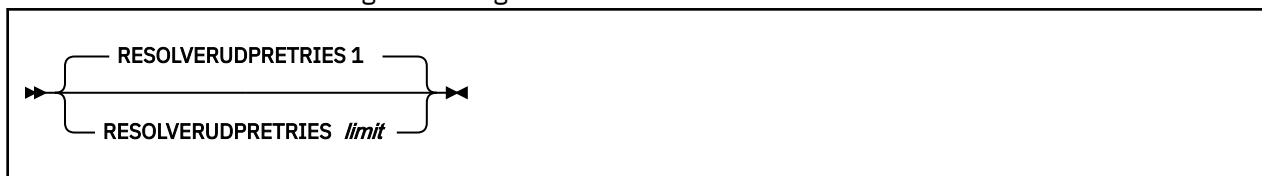
timeoutvalue

Indicates the number of seconds the resolver waits until a response is received. The default open time-out is 30 seconds.

RESOLVERUDPRETRIES statement

Purpose

The RESOLVERUDPRETRIES statement specifies the number of times the resolver should try to connect to the name server when using UDP datagrams.



Operands

limit

Indicates the maximum number of times the resolver should try to connect to the name server. The default is 1.

RESOLVEVIA statement

Purpose

The RESOLVEVIA statement specifies the protocol used by the resolver to communicate with the name server.



Operands

TCP

Specifies that the protocol is TCP.

UDP

Specifies that the protocol is UDP. The default protocol is UDP.

SECURETELNETCLIENT statement

Purpose

The SECURETELNETCLIENT statement provides the default client security value to use when neither the SECURE nor NOSECURE option is specified on the TELNET command. Specifying a client security option on the TELNET command overrides the SECURETELNETCLIENT value. The values for the SECURETELNETCLIENT statement correspond to the NOSECURE and SECURE options on the TELNET command.



Operands

SECURETELNETCLIENT YES

A secure TELNET connection will be attempted if the TELNET server supports secure connections. If the TELNET server does not support secure connections, or if the SSL server is not running on the local system, the TELNET command will fail. Specifying the NOSECURE option on the TELNET command overrides this value.

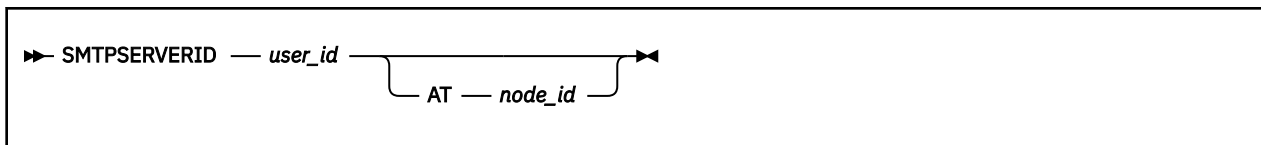
SECURETELNETCLIENT NO

A clear TELNET connection will be attempted. If the TELNET server does not support clear connections, the TELNET command fails. This is the default. Specifying the SECURE option on the TELNET command overrides this value.

SMTPSERVERID statement

Purpose

The SMTPSERVERID statement identifies the virtual machine that provides SMTP services, if one exists.



Operands

user_id

Specifies the 1- to 8-character user ID of the SMTP server virtual machine for either the local system, or for a network gateway system.

AT node_id

Specifies the 1- to 8-character RSCS node ID of the location of the SMTP server gateway. The local node is used when a node ID is not specified.

Usage Notes

Multiple `SMTPSERVERID` statements can be specified to identify multiple SMTP servers in your environment. However, only the first instance is used by those CMS functions that use the `SMTPSERVERID` definition to determine a user ID to which SMTP-directed data should be directed.

TCPIPUSERID statement

Purpose

The `TCPIPUSERID` statement specifies the user ID of the TCPIP virtual machine.



Operands

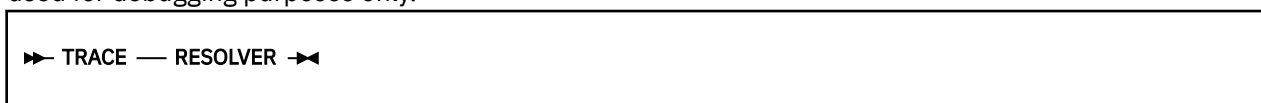
user_id

Specifies the 1- to 8-character user ID of the TCPIP virtual machine. TCPIP is the default user ID.

TRACE RESOLVER statement

Purpose

If specified, the `TRACE RESOLVER` statement causes a complete trace of all queries to and responses from the name server to be written to the user's console. The `TRACE RESOLVER` statement should be used for debugging purposes only.



Operands

The `TRACE RESOLVER` statement has no operands.

UFTSERVERID statement

Purpose

The UFTSERVERID statement identifies the virtual machine that provides UFT services, if one exists.

➤ UFTSERVERID — *user_id* ➤

Operands

user_id

Specifies the 1- to 8-character user ID of the UFT server virtual machine for the local system. A user ID of asterisk (*) has special meaning and indicates the RSCS virtual machine ID that is returned by the CMS IDENTIFY command should be used.

Usage Notes

Multiple UFTSERVERID statements can be specified to identify multiple UFT servers in your environment. However, the CMS SENDFILE command assumes the first instance defines the UFTASYNCR UFT server.

USERDATA statement

Purpose

The USERDATA statement marks the beginning of user-defined parameters that are defined in the TCPIP DATA file.

➤ USERDATA — *parameter* — ENDUSERDATA ➤

Operands

parameter

Specifies an arbitrary, user-defined value.

Any parameters on the USERDATA statement are ignored by TCP/IP clients that are included as part of the TCP/IP for VM feature.

VMFILETYPE statement

Purpose

The VMFILETYPE statement defines translation and line values to be associated with a specific file *extension* (file type). These values are used by the NFS client, as well as the FTP and NFS servers, when a file with this extension is processed.

➤ VMFILETYPE — *extension* — { ,TRANSLATE=NO ¹ ,TRANSLATE=YES } { ,LINES=NONE ¹ ,LINES=NL ² CMS } ➤

Notes:

- ¹ Default when no VMFILETYPEDEFAULT statement exists.
- ² Used by only the NFS server.

Note: Do not include intervening spaces with the **TRANSLATE=** or **LINES=** parameters when these are specified.

Operands

extension

Defines a file type extension for which translation and line values are being defined; *extension* is a required parameter. The specified file type can begin or end with an asterisk (*) to "wildcard" that file type.

TRANSLATE=YES

TRANSLATE=NO

Indicates whether EBCDIC-ASCII translation is performed when files with the specified *extension* are processed. Specify TRANSLATE=YES if EBCDIC-ASCII translation is to be performed, or TRANSLATE=NO if translation should not be performed.

The TRANSLATE= parameter is optional. If this parameter is omitted, the default established by the VMFILETYPEDEFAULT statement is used; if no such statement exists, TRANSLATE=NO is assumed. For more information, see [Chapter 19, "Using Translation Tables,"](#) on page 667.

LINES=NL

LINES=NONE

LINES=CMS

Indicates whether line feed characters are inserted at CMS record boundaries when files with the specified *extension* are processed. This parameter is used by only the NFS server.

Specify LINES=NL, if line feed characters are to be inserted at CMS record boundaries.

If LINES=NONE is specified, no line feed characters are inserted.

To maintain CMS file format, specify LINES=CMS. When CMS variable-length record files are processed, a binary, unsigned, two-byte length field is visible at the beginning of CMS records.

The LINES= parameter is optional. If this parameter is omitted, the default established by the VMFILETYPEDEFAULT statement is used; if no such statement exists, LINES=NONE is assumed.

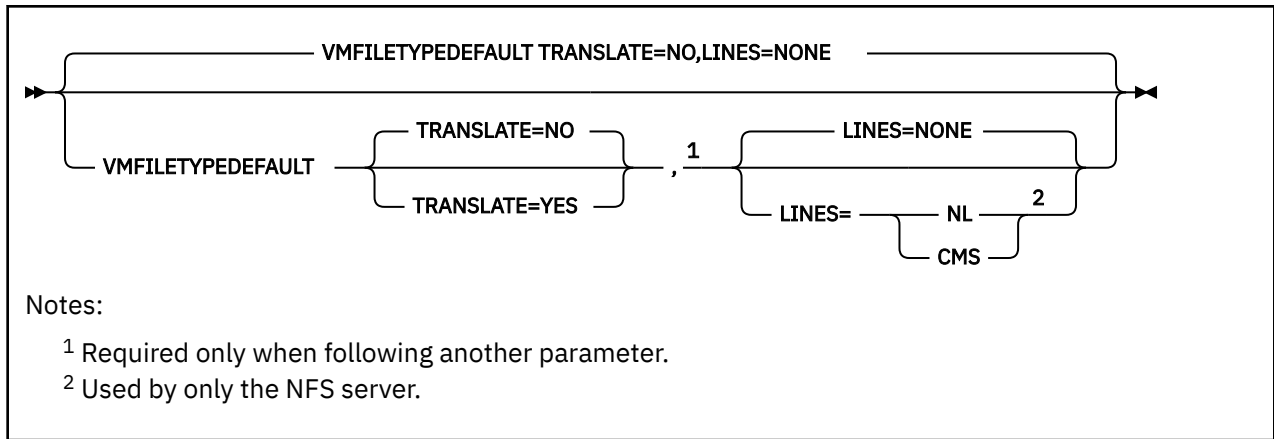
Usage Notes

When the VM NFS client, NFS server, or FTP server compares the *filetype* of a file being processed with those defined by a VMFILETYPE statement, case (upper or lower) is ignored. For example, BIN is considered to be equivalent to Bin.

VMFILETYPEDEFAULT statement

Purpose

The VMFILETYPEDEFAULT statement defines translation and line values to be applied to file extensions for which no VMFILETYPE statements are defined. These default values are used by the NFS client, as well as the FTP and NFS servers, when such files are processed.



Note: Do not include intervening spaces with the TRANSLATE= or LINES= parameters when these are specified.

Operands

TRANSLATE=YES

TRANSLATE=NO

TRANSLATE=NO

Indicates whether EBCDIC-ASCII translation is performed when a file is processed. Specify TRANSLATE=YES if EBCDIC-ASCII translation is to be performed, or TRANSLATE=NO if translation should not be performed.

The TRANSLATE= parameter is optional. If this parameter is omitted, TRANSLATE=NO is assumed. For more information, see [Chapter 19, "Using Translation Tables,"](#) on page 667.

LINES=NL

LINES=NONE

LINES=CMS

Indicates whether line feed characters are inserted at CMS record boundaries when files are processed. This parameter is used by only the NFS server.

Specify LINES=NL if line feed characters are to be inserted at CMS record boundaries.

If LINES=NONE is specified, no line feed characters are inserted.

To maintain CMS file format, specify LINES=CMS. When CMS variable-length record files are processed, a binary, unsigned, two-byte length field is visible at the beginning of CMS records.

The LINES= parameter is optional. If this parameter is omitted, LINES=NONE is assumed.

Testing the TCP/IP System Configuration

Use the HOMETEST command for testing the TCP/IP system configuration.

HOMETEST Command

➤ HOMETEST ➤

Purpose

The HOMETEST command can test the system configuration including HOSTNAME, DOMAINORIGIN, and NSINTERADDR, which are defined in the TCPIP DATA file.

Defining System Parameters

HOMETEST verifies that the host tables or name server, depending on the NSINTERADDR statement, can resolve the fully qualified domain name (defined by HOSTNAME and DOMAINORIGIN statements) for your site. In addition, the internet addresses corresponding to your site HOSTNAME are checked against the HOME list. This is defined in the PROFILE TCPIP file. A warning message is issued if any addresses are missing from the HOME list.

Usage Notes

1. Verify that the TCPIP virtual machine has been started before you use the HOMETEST command.

Chapter 4. Configuring the Local Host Files

The local host files contain information needed for local host name resolution. Any domain name or IP address specified in this file is accessible for use on your network. Local host files are used to create the site table, which enables name resolution and reverse name resolution without using a domain name server.

TCP/IP for z/VM offers two local host files for domain name resolution and reverse name resolution. The old HOSTS LOCAL file (which supports IPv4 only), and the preferred ETC HOSTS file (which supports both IPv4 and IPv6).

The ETC HOSTS file does not require additional processing to create the site tables used for name resolution. The site tables are created dynamically by the resolver when the ETC HOSTS file is used. Use of the HOSTS LOCAL file requires that you run the MAKESITE command to create the site tables. Whenever changes are made to the HOSTS LOCAL file, you must run the MAKESITE command to recreate the site tables.

A sample file, ETCHOSTS SAMPLE, is supplied with the z/VM system deliverable on the TCPMAINT 592 disk. You can use this file as a guide for creating a customized ETC HOSTS file, that should reside on this same minidisk (TCPMAINT 592). Because each site is unique, the statements within the ETC HOSTS file must be customized for your installation.

The HOSTS SLOCAL sample file is also supplied on the TCPMAINT 592 disk and can be copied to the TCPMAINT 198 disk and customized for your site, but HOSTS LOCAL processing is not the preferred method for name resolution.

If you have a HOSTS LOCAL file and want to convert the file information into ETC HOSTS format, use the sample exec, LCL2ETC SAMPEXEC, supplied on the TCPMAINT 592 disk.

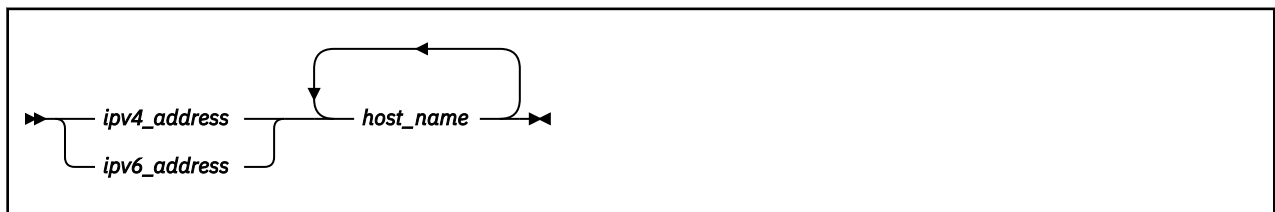
When local host files are used for name resolution (instead of using a nameserver), TCP/IP for z/VM first determines whether an ETC HOSTS file is available. If an ETC HOSTS file is available, TCP/IP for z/VM uses it for name resolution. If no ETC HOSTS file is present, TCP/IP for z/VM will attempt to use the local site tables created by the MAKESITE command.

The method used to resolve host names (local files or domain name server) and in what order these methods are used are specified using the DOMAINLOOKUP statement in the TCPIP DATA configuration file.

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

ETC HOSTS File Syntax

Entries in the ETC HOSTS file use the following statement syntax:



Operands

ipv4_address

The IPv4 address of the host.

ipv6_address

The IPv6 address of the host.

host_name

The host name associated with the specified IP address. The *host_name* operand can be a maximum of 128 characters and must contain one or more tokens separated by a period. Each token must be larger than one character and less than 64 characters. Also, the first character in each token must start with a letter (A-Z or a-z).

Usage Notes

- The maximum line length supported is 256 characters. If a line is greater than 256 characters, it is truncated to 256 characters and processed. If trace resolver is active, a warning message is issued.
- The ETC HOSTS file can contain IPv4 and IPv6 addresses, but IPv4 mapped addresses are not supported. Each IP address can have up to 35 host names.
- A comment is indicated by the # or ; character.
- If you need loopback support, you must explicitly code a LOOPBACK entry for address 127.0.0.1 in the ETC HOSTS file.

HOSTS LOCAL File Syntax

The HOSTS LOCAL file can contain two types of entries:

- Host Statement
- Network Statement

One line of the HOSTS LOCAL file is used for each distinct host and ends with four colons. Each line has three essential fields, separated by colons:

- The keyword HOST or NET
- A comma-separated list of internet addresses for that host
- A comma-separated list of fully qualified names for that host.

Host entries also have three optional fields.

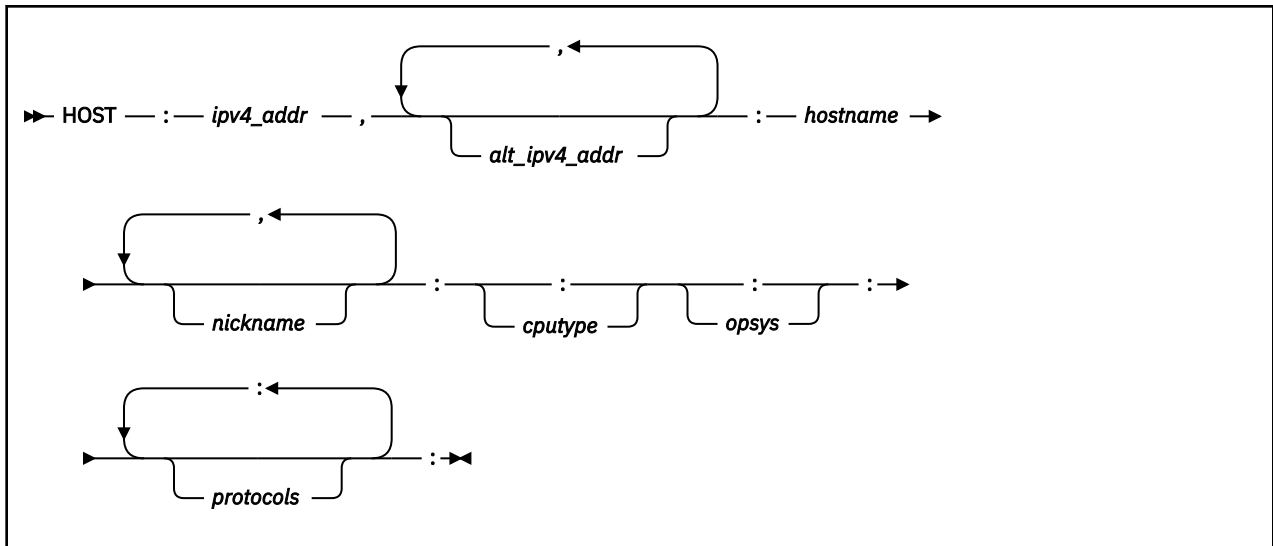
Example: If you have two local hosts, LOCAL1 (internet addresses 192.6.77.4 and 192.8.4.1) and LOCAL2 (with an alias LOCALB and internet address 192.6.77.2), append the following lines to the HOSTS LOCAL file:

```
HOST :192.6.77.4, 192.8.4.1 :LOCAL1 :::  
HOST : 192.6.77.2 : LOCAL2, LOCALB :::
```

Note: The maximum length for a host allowed in the HOST tables is 24 characters. Names longer than this limit are truncated without warning. However, the name server does not have a maximum character length.

HOST Statement

The HOST statement specifies the address and network name of a host that is accessible to TCP/IP.



Operands

ipv4_addr

(IPv4 only) Specifies the internet address of the host.

alt_ipv4_addr

(IPv4 only) Specifies the alternative internet address of the host. A maximum of 6 alternative internet addresses may be specified for each host.

hostname

Specifies the fully qualified name of the host. The name can be a maximum of 24 characters and can include the minus sign (-) and a period (.). Periods can only be used to delimit components of a domain name. No blank or space characters are allowed. The *hostname* is not case sensitive. The first character must be an alpha character and the last character cannot be a minus or period.

nickname

Specifies the fully qualified nickname of the host.

cputype

Specifies the machine type or processor type.

opsys

Specifies the operating system of the host.

protocols

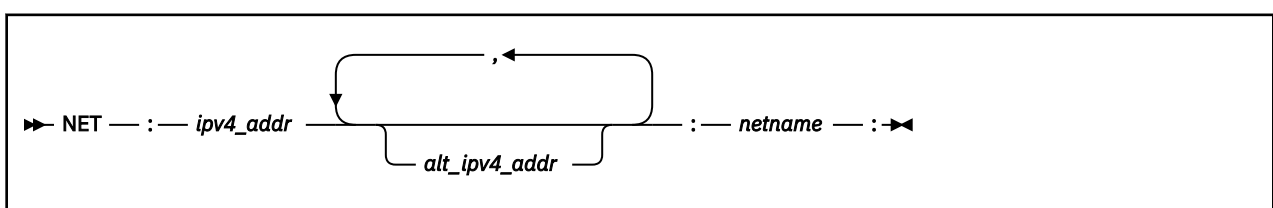
Specifies the protocols that are supported by the host.

NET Statement

Purpose

The NET statement specifies the address and network name of the local network that is accessible to TCP/IP.

Note: The NET statement is not used by IBM-provided TCP/IP applications. It is provided for those applications that use the `getnetent()` socket call.



Parameters

ipv4_addr
(IPv4 only) Specifies the internet address of the network.

alt_ipv4_addr
(IPv4 only) Specifies the alternative internet address of the network.

netname
Specifies the fully qualified name of the network. The name can be a maximum of 24 characters and can include the minus sign (-) and a period (.). Periods can only be used to delimit components of a domain name. No blank or space characters are allowed. The *netname* is not case sensitive. The first character must be an alpha character and the last character cannot be a minus or period.

Building the HOSTS LOCAL Site Table

When you initially create or make changes to your HOSTS LOCAL file, you must generate and install new HOSTS SITEINFO and HOSTS ADDRINFO files. These files cause the fastest look-up by creating a hash table of all entries in the HOSTS LOCAL file.

Before you begin: You need to access to the TCPMAINT user ID and the TCPMAINT 198 disk.

Perform the following steps to generate the HOSTS SITEINFO and HOSTS ADDRINFO files:

- 1. Log on as TCPMAINT.
- 2. Modify the HOSTS LOCAL file on TCPMAINT 198. If this is the first time you are modifying the HOSTS LOCAL file, you can copy the sample file, HOSTS SLOCAL, from TCPMAINT 592 to TCPMAINT 198.

Note: The HOSTS LOCAL table may not contain more than 199,200 site names or 69,500 IP addresses. Sites that require more entries should consider using Domain Name Server (DNS) services.

- 3. Run the MAKESITE program to produce the HOSTS SITEINFO and HOSTS ADDRINFO files from the newly modified HOSTS LOCAL file. The virtual machine size required to process a HOSTS LOCAL file with the maximum entries must be large enough to allocate approximately 15M of contiguous storage.



Parameter	Description

fm
The file mode at which the HOSTS SITEINFO and HOSTS ADDRINFO files are to be created. **A** is the default.

WARN
Indicates that additional warning messages should be presented during processing, as warranted. Output files are produced at file mode A when this operand is used.

Note: If you are processing the maximum number of entries, it is recommended that you have defined a *large* virtual machine, 64M. This can take a significant amount of time (up to two hours) to complete this task.

- 4. If the HOSTS SITEINFO and HOSTS ADDRINFO files exist on the TCPMAINT 592 disk, rename them, for example, to HOSTS SITEOLD and HOSTS ADDROLD. These files are currently accessed by other TCPIP users on the system, and should not be deleted immediately.
- 5. Copy the new HOSTS ADDRINFO and HOSTS SITEINFO files from your A disk to the TCPMAINT 592 disk.

6. Run the TESTSITE program to test the correctness of the HOSTS SITEINFO file. This step is optional.

▶▶ TESTSITE ◀◀

When prompted for a name, enter the name you want to verify. When you have checked all the names in question, enter QUIT.

You are done when you have checked all the names with TESTSITE.

Chapter 5. General TCP/IP Server Configuration

This chapter describes the virtual machines and methods of server configuration for TCP/IP.

Virtual Machine Definitions

This section describes the virtual machines that are necessary to provide basic and optional TCP/IP services. The virtual machines listed here comprise a set of “default” TCP/IP virtual machines that are defined as part of the z/VM system when it is installed.

Additional or differently-named virtual machines can be defined for your system to provide (or augment) the function provided by the “default” servers discussed in this section. For more information about defining such virtual machines and any applicable machine-specific requirements, see the *Program Directory for TCP/IP for z/VM*.

While various TCP/IP virtual machines have specific definition requirements, all TCP/IP servers must maintain links to the following minidisks, to allow for correct operation:

Table 3. Required TCP/IP Server Minidisk Links

Minidisk	Description
TCPMAINT 592	Client-code disk
TCPMAINT 591	Server-code disk
TCPMAINT 198	Configuration file, or <i>customization</i> disk

Note that much of the installation process for TCP/IP is completed as part of the installation of z/VM itself, so the “default” TCP/IP virtual machines described in this section, and the minidisk (or, DASD) resources they require, will have already been defined for your system. Thus, the directory definitions (supplied with z/VM) for the “default” TCP/IP virtual machines include links for the minidisks listed in [Table 3 on page 33](#).

Samples for many commonly-used TCP/IP configuration files have been supplied and placed on the appropriate (production) minidisks, but you must customize the samples for your environment. These files are discussed further in [“TCP/IP Configuration File Overview” on page 50](#).

Required Virtual Machines

The following virtual machines are required to provide basic TCP/IP services:

Table 4. Required Virtual Machines

Machine	Function
7VMTCP30	Maintains the TCP/IP system. Installation and service resources are owned by this user ID.
TCPIP	Provides TCP/IP communication services. The Telnet server is implemented as an “internal client” within the TCPIP virtual machine.
TCPMAINT	Owns TCP/IP production resources — the 198, 591, and 592 disks.

Optional Virtual Machines

The following table lists optional virtual machines; these servers provide optional TCP/IP services:

Table 5. Optional Virtual Machines

Machine	Function	Location
CSMSERVE	Provides restricted File Transfer Protocol (FTP) server support for z/VM Centralized Service Management (z/VM CSM) controlled access to files on a local VM host.	Chapter 6, “Configuring the FTP Server,” on page 55
FTPSERVE	Provides FTP server support for controlled access to files on the local VM host.	Chapter 6, “Configuring the FTP Server,” on page 55
GSKADMIN	Owns the BFS file space in which the SSL key database resides, and serves as database management user ID. Also owns the BFS file space used as SSL server temporary work space, and serves as an SSL server administrative user ID.	Chapter 15, “Configuring the SSL Server,” on page 453
LDAPSRV	Provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange.	Chapter 7, “Configuring the LDAP Server,” on page 79
MPROUTE	The MPRoute server implements the OSPF, IPv6 OSPF, RIP, or IPv6 RIP protocols (or any combination of those protocols), and provides an alternative to static gateway definitions.	Chapter 8, “Configuring the MPRoute Server,” on page 193
PORTMAP	Runs the Portmapper function for RPC on systems that support the Network File System protocol.	Chapter 10, “Configuring the Portmapper Server,” on page 347
REXECD	The virtual machine for the REXECD server. It provides a remote execution service machine for TCP/IP hosts that support the REXEC client.	Chapter 11, “Configuring the REXEC Server,” on page 349
RXAGENT n	The agent virtual machines used by REXECD to execute commands from anonymous rexec clients.	Chapter 11, “Configuring the REXEC Server,” on page 349
SMTP	The virtual machine for the SMTP server. It receives mail over a TCP/IP network connection or from its virtual reader, and then sends that mail through the TCP/IP or RSCS network, according to the mail destinations.	Chapter 13, “Configuring the SMTP Server,” on page 375
SNMPD	The SNMP Agent virtual machine.	Chapter 14, “Configuring the SNMP Servers,” on page 439
SNMPQE	The SNMP Query Engine virtual machine.	Chapter 14, “Configuring the SNMP Servers,” on page 439
SSLDCCSM	Creates and owns the Discontiguous Saved Segment (DCSS), used by an SSL server pool to maintain session cache information.	“The SSL DCSS Management Agent Virtual Machine” on page 463
SSL $nnnn$	An SSL “pool” server virtual machine.	“The SSL Pool Server Virtual Machine” on page 463

Table 5. Optional Virtual Machines (continued)

Machine	Function	Location
UFTD	Implements the Unsolicited File Transfer (UFT) server.	Chapter 17, “Configuring the UFT Server,” on page 649
VMNFS	Implements the Network File System (NFS) server.	Chapter 9, “Configuring the NFS Server,” on page 325

Methods of Server Configuration

This section describes server configuration methods that allow you to add servers and server classes or modify the characteristics of a specific TCP/IP server virtual machine. Duplication of existing servers is also described. Configuration changes such as these are generally accomplished using the DTCPARMS file, described in [“The DTCPARMS File” on page 35](#). Note that the DTCPARMS file is used only to configure CMS servers; GCS servers are configured using a *userid* GCS file, as explained in [“GCS Servers” on page 49](#).

In general, the DTCPARMS file allows you to define operational aspects for TCP/IP servers that are related to their operation within a z/VM environment, such as ensuring the correct run-time environment or virtual machine attributes.

Protocol- or application-specific configuration support is provided through one or more server-specific configuration files. These files are listed in summary form in [“TCP/IP Configuration File Overview” on page 50](#).

The DTCPARMS File

Configuration of each server is controlled by a set of files with a file type of **DTCPARMS**. These files may contain two types of information:

1. Server *class* names that define the application protocols available for all server virtual machines.
2. Individual server user IDs and their associated server class, as well as the operational characteristics of the server (security, devices, parameters, etc.).

The TCP/IP server initialization program searches for server definitions in a hierarchical fashion. The following table lists the DTCPARMS files in the order that they are searched, along with a description of each file.

Table 6. DTCPARMS File Search

File	Purpose
<i>userid</i> DTCPARMS	Can be used for servers that do not require configuration by the TCP/IP administrator, such as a test server. Such a file might commonly reside on a server disk or directory accessed at file mode A.
<i>nodeid</i> DTCPARMS	Useful for shared-DASD configurations. The node ID used is the node ID returned by the CMS IDENTIFY command. This file should be maintained on the TCPMAINT 198 disk.
SYSTEM DTCPARMS	Most customized server configurations should be maintained in this file. This file should be maintained on the TCPMAINT 198 disk.
IBM DTCPARMS	Server classes provided by IBM, as well as the default server configurations, are supplied by this file. This file resides on the TCPMAINT 591 disk and should never be modified, because it can be replaced when service is applied, or when a new release is installed. All modifications required for your installation should be placed in SYSTEM DTCPARMS (or, <i>nodeID</i> DTCPARMS, as warranted).

In general, the entries for individual servers need not reside in the same file as the server class they reference. For example, the server entry for user ID FTPSERV2 can be in SYSTEM DTCPARMS, while its server class, **ftp**, might be in IBM DTCPARMS. However, for some configurations — such as those used for secure communications support — certain entries can be required to reside within a certain set of DTCPARMS files.

Entries for individual servers do not have to be in the same file as the server class they reference. For example, the server entry for user ID FTPSERVE may be in SYSTEM DTCPARMS, but its server class, **ftp**, may be in IBM DTCPARMS.

In addition to the DTCPARMS files, server configuration information can be provided by server *profile exits*. See “[Server Profile Exits](#)” on page 46 for information.

Configuring the DTCPARMS File

Before configuring the DTCPARMS file, you should be familiar with the format, usage information, and tags used in this file. This information is described in the following sections.

DTCPARMS File Format

The DTCPARMS file uses a format similar to CMS NAMES files and is maintained using XEDIT. Two types of entries comprise this file: `server` definitions that identify specific server virtual machines, and `class` definitions that define specific attributes to support the application protocol used by a given server.

The following sample entries define the configuration for the TCPIP virtual machine:

```
:Nick.TCPIP :Type.server :Class.stack
:Nick.stack :Type.class
           :Name.TCP/IP Stack
           :Command.TCPIP
           :Runtime.PASCAL
           :Diskwarn.YES
           :Authlog.AUTHLOG FILE A
```

The `:Nick.TCPIP` entry defines the TCPIP user ID as a *server* entry type; this server is an instance of the **stack** server *class*. The `:Nick.Stack` entry defines the attributes and characteristics of this class.

When entries are defined or modified, keep the following in mind:

- Entries consist of *tags* and *tag values*.
- Entries that define a server using a `:Type.Server` definition must also include a `:Class.` tag and value to identify the *class* to which that server belongs.
- Tags defined as part of a *server* entry will be used for only that server instance (that is, the specific virtual machine user ID identified by the `:Nick.` tag).
- Tags defined as part of a *class* entry will be used for all servers of that class (unless overriding tags are defined as part of a *server* entry that references the class).
- If multiple tags have the same name, the last one is used.
- Uppercase and lowercase characters can be used interchangeably for tags and most of their values.
- All tags do not apply to all servers.
- Any tag not recognized is ignored without warning.
- Every entry must include either a `:Type.Server` or a `:Type.Class` definition.
- Values cannot start with a colon, end with a period, or otherwise have the syntax of a tag.
- Certain `:Type.Server` entries might be required to be defined within a *nodeid* or SYSTEM DTCPARMS file, to allow for cross-referencing by servers of different classes.
- A DTCPARMS tag name starts with a `:` (colon) and ends with a `.` (period). It can contain any alphanumeric characters, plus `_` (underscores).

- Valid DTCPARMS file comment delimiters are the same as those defined for use in a CMS NAMES file, as documented by the CMS NAMEFIND command. These delimiters are: * (asterisk) and . * (period plus asterisk).
- Not all erroneous DTCPARMS entries can be identified or isolated at runtime, for which TCP/IP server initialization may still occur.

DTCPARMS Tags

The following table lists DTCPARMS file tags that can be used when configuring servers. Descriptions of these tags are also included.

Table 7. DTCPARMS Tags for Configuring Servers

Tag	Description
:Admin_ID_List.	A list of user IDs that are to be included in a private resource registration file (\$SERVER\$ NAMES, created or replaced during server initialization). The listed user IDs are authorized to interact with the subject server using queue-based interprocess communication (IPC), for server administrative or similarly restricted purposes. <pre>:admin_id_list.TCPMAINT MAINT OPERATOR SYSADMIN</pre> There is no default value for this tag. Applies to the ssl class only.
:Anonymous.YES :Anonymous.NO	A value of YES will enable access to the server without requiring a VM user ID and password. The default is NO. Applies to the nfs, ftp, and rexec classes only.
:Attach.raddr (option), raddr AS vaddr (option), raddr1-raddr2 (option)	A list of real addresses to be attached to this server. This server must have IBM privilege class B. Devices may be specified individually or as a range. Multiple devices or ranges must be separated by commas. A virtual address may be specified for a real address (not an address range) by appending "AS" followed by the virtual address. (option) can be specified as either: • REquired: indicates that <i>raddr</i> is a required device. If a problem is encountered during the attach of this device, server initialization is terminated. This is the default. • OPTional: indicates that <i>raddr</i> is an optional device. If a problem is encountered during the attach of this device, an error message is displayed and server initialization continues. If <i>option</i> is specified, it must be enclosed within parentheses. Example: <pre>:attach.1500 AS 500, 400-403 (req), 800-803</pre> Applies to the stack class only.
:Authlog.file_ID	The name of the file used by the TCP/IP server for logging unauthorized attempts to use TCP/IP services. The default is TCPIP AUTHLOG A. Applies to the stack class only.

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:Class.server_class	Identifies the TCP/IP application protocol used by this server. Select from IBM-provided values or specify a protocol that has been defined by a :type.class entry. IBM-provided values: ftp, ldap, mproute, nfs, portmapper, rexec_agent, rexec, smtp, snmp, snmpqe, ssl, stack, uftd.
:Command.command	The command to run for this server.
:Config.file_ID	Identifies the name of the configuration file the MPRoute server uses when it initializes. The server treats a file mode specification (if included) as if an asterisk (*) had been specified; that is, the server uses the first configuration file it finds according to the CMS search order. Applies to the mproute class only.
:CSLibs.csllib-list	CSL libraries to be added to those implicitly defined by :runtime.
:DB2_Database.dbname	Identifies the Db2 database used by this server. SQLINIT will be issued for this database as well as a SET LANGUAGE (ADD ARI USER). Should be accompanied by the :vmlink. tag to access the Db2 run disk. For the dns class, do not specify a value for this tag if the domain name server is running in caching-only mode, which is the default.

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:DCSS_Parms.<NONE> :DCSS_Parms.<DEFAULT> :DCSS_Parms.dcssname hexpage1-hexpage2	Identifies parameters that are used to define the restricted Discontiguous Saved Segment (DCSS), which facilitates SSL server maintenance (and as appropriate, sharing) of session cache information. The default is <NONE>, which indicates the subject TCP/IP server is not configured for secure communications support. <NONE> negates logon control processing of an SSL DCSS management agent server and the definition of a SSL shared session cache DCSS. The value DEFAULT signifies that the subject TCP/IP server is configured for secure communications support, for which logon control processing of an SSL DCSS management agent server is performed, and from which an SSL shared session cache DCSS is defined. The DEFAULT and <i>dcssname hexpage1-hexpage2</i> values signify that the subject TCP/IP server is configured for secure communications support, for which logon control processing of an SSL DCSS management agent server is performed, and from which an SSL shared session cache DCSS is defined. When DEFAULT is specified, the value of the <code>:stack.</code> tag defined for this agent server (or, its default of TCPIP) is employed as the DCSS name, along with system-defined page range values. The value DEFAULT should be specified unless installation requirements dictate otherwise. When circumstances require the use of a <i>dcssname hexpage1-hexpage2</i> value, the supplied parameters must conform to requirements imposed by the CP DEFSEG command (that is, the DCSS name must be a 1 to 8 character alphanumeric value, and the given page range values must be appropriate for the segment definition). In addition, the <i>dcssname</i> value (whether specified directly, or when a stack user ID default is employed) must match the segment name specified for a NAMESAVE statement in both: - the CP directory entry of an applicable SSL DCSS management agent virtual machine - the profile entry for the pertinent SSL server pool. Note that no facilities exist to confirm that the DTCPARMS <i>:DCSS_Parms.dcssname</i> value and any CP directory profile NAMESAVE statement values match. When used, this <i>:DCSS_Parms.</i> tag must be defined as part of a <code>:type.server</code> entry, in a <i>nodeid</i> or SYSTEM DTCPARMS file. Applies to the stack class, but also is cross-referenced by ssl and ssl_dcss_agent class servers.
:Diskwarn.nn% :Diskwarn.nn%-fm :Diskwarn.YES :Diskwarn.YES-fm :Diskwarn.NO	A warning is issued if less than nn% of read/write space is available on the indicated file mode. A value of YES only verifies that the file mode is accessed read/write. The percent sign is optional. The default is NO. If no file mode is specified, file mode A is the default. Note: While it is possible to disable the warning for any disk attached to a server, some servers have specific requirements for Read/Write minidisks. For more information, see “User ID Minidisk Considerations” on page 5.

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:ESM_Enable.YES :ESM_Enable.NO	The External Security Manager (ESM) will be used to authenticate and authorize access to resources managed by this server. The default is NO. For more information, see Appendix A, “Using TCP/IP with an External Security Manager,” on page 685.
:ESM_Racroute.YES :ESM_Racroute.NO :ESM_Racroute.command	The command to be used for initialization and termination of a RACROUTE environment. A value of YES causes the command RPIUCMS to be used. A value of NO indicates that no initialization of the RACROUTE environment is to be performed. The use of RACROUTE by IBM-provided server applications is described in Appendix A, “Using TCP/IP with an External Security Manager,” on page 685. The default is NO.
:ESM_Validate.YES :ESM_Validate.NO :ESM_Validate.filename	The name of a module to be used to validate user IDs and passwords supplied by clients when the ESM does not support Diagnose code X'88' subcode 8. This module is preloaded into memory for improved performance. A value of YES causes RPIVAL MODULE to be used. A value of NO prevents loading of any module - RPIVAL module will be read from disk if :ESM_Enable.YES is specified. The default is NO.
:Exit.exec-name	The name of an exec to be run according to the rules defined in “Server Profile Exits” on page 46. The output from this exec overrides any specification returned by the global exit, TCPRUNXT EXEC.
:For.userid	The VM user ID of the TCP/IP protocol server on whose behalf this server will do work. There is no default. Applies to the rexecd_agent and ssl_dcscs_agent class servers.
:Loadlibs.loadlib-list	CMS LOADLIBS to be added to those implicitly defined by :runtime.
:Memory.nnnnK :Memory.nnnnM	The minimum virtual storage size required to run server. If the virtual machine is not large enough, an attempt will be made to define more storage and re-IPL CMS. If not specified, no check is performed.
:Mixedcaseparms.	Designates that mixed case is to be preserved for startup parameters defined using the :Parms. tag.
:Mount.bfs-pathname [mount-point-pathname] [(OPENVM-MOUNT-options) [, bfs-pathname [mount-point-pathname] [(OPENVM-MOUNT-options) ...]	The fully-qualified Byte File System directory to be mounted. If a path name for the mount point is not provided, the specified BFS directory is mounted as the root directory (/). OPENVM MOUNT command options, if included, must be preceded by a left parenthesis. You can define multiple mounts by specifying additional mount parameter groups (bfs-pathname [mount-point-pathname] [(OPENVM-MOUNT-options) groups), with each such group separated by a comma. Values defined for this tag are case sensitive. The path name cannot include commas or blanks. If commas or blanks are required, use the :Exit. tag to identify a server profile that contains OPENVM MOUNT commands to mount the desired directories. If the <i>filespaceid</i> portion of a fully-qualified BFS path name specified for this tag is omitted, the user ID of the server is substituted as the file space ID. For example: <pre>/ . . /VMBFS:VMSYS: /</pre>

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:Name.description	Provides a description of the server to be displayed when the server is started. Mix cased values for this tag can be preserved.
:Nick.userid :Nick.prefix* :Nick.class_name	For :type.server , the VM user ID of a specific TCP/IP server, or a wildcard user ID prefix for an SSL server pool. For :type.class , an arbitrary name to be referenced by a :class. tag. When a wildcard <i>prefix*</i> value is specified, the following restrictions apply: • This entry must be defined within a DTCPARMS file that can be referenced by multiple server classes. That is, within a <i>nodeID</i> or SYSTEM DTCPARMS file. • The specified prefix value must match that of a 1 to 3 character user ID, defined with the POOL statement in the CP directory. • Applies to the ssl class only, but also is cross-referenced by stack class servers.
:OCSPParms.parameters	Used to configure the OCSP- and CDP-related information if you are using OCSP or CDP certificate revocation checking or both. See “The :OCSPParms. Tag” on page 459 for more information.
:Owner.userid	User ID to receive the console log. If an invalid user ID is specified, the log will be kept in the server's virtual reader. The default is TCPMAINT .
:Parms.parameters	Defines startup parameters to be passed to the server. Parameters that affect the security characteristics of a server are automatically generated through the use of the :Anonymous. and :ESM_Enable. tags. Therefore, these parameters should not be specified using the :Parms. tag. See “Automatic Generation of Selected Startup Parameters” on page 44 for more information about parameters to which this applies. Parameters provided through the use of the :Parms. tag may override those that are generated automatically. When Language Environment run-time options are specified with server program parameters (or, arguments), all run-time options must precede any program parameters, and must be separated from these by a blank-delimited slash (/).
:Runtime.C :Runtime.PASCAL	C establishes the Language Environment for VM and MVS (SCEERUN LOADLIB). PASCAL establishes the VS PASCAL 1.2 runtime environment (TCPRTLIB LOADLIB). If this tag is not specified, a specific runtime language environment is not established.
:Stack.stackID	If specified, the TCP/IP stack user ID (<i>stackID</i>) is compared to that cited by the TCPIPUSERID statement in the TCP/IP data file (TCPIP DATA, by default), or an equivalent such file, as designated by a :TcpDataFile. tag. If the compared user ID values do not match, the server is not started. For ssl and ssl_dcsc_agent class servers, this tag also allows such servers to be identified as being associated with a given TCP/IP stack server, when it has been configured to support secure communications.

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:TcpDataFile. <i>fname ftype</i>	If specified, identifies a particular TCP/IP data file that is to be referenced when the subject server is initialized. When not specified, the first available instance of the file TCPIP DATA is referenced by default. If a file type (<i>ftype</i>) is not supplied for the specified value, 'DATA' is used by default. The file cited by this tag is copied to the minidisk or directory accessed at CMS filemode A, to avoid potential content conflicts with a TCPIP DATA file that might exist elsewhere in the CMS search order of the server. The copied instance is deleted at completion of a normal server shutdown.
:Timestamp.ON :Timestamp.OFF :Timestamp.CP	Controls whether server console messages and command responses are prefixed with the local time of day. Recognized tag values are those for the CP TERMINAL TIMESTAMP command (ON, OFF or CP). When this tag is specified, the given value is applied during server initialization. An invalid value is ignored without notification, and the value OFF is employed. When this tag (or, its value) is omitted, any existing CP TERMINAL TIMESTAMP setting remains in effect. There is no default value for this tag.
:Timezone.stdoffset[<i>dst[offset]</i> [, <i>rule</i>]]	Sets the time zone for the server. For detailed descriptions of the operands, see the tzset() function description in XL C/C++ for z/VM: Runtime Library Reference. Example: <pre>:Timezone.EST5EDT</pre> Applies to the mproute, nfs and ssl classes only.
:Type.class :Type.server	Class identifies this entry as an application protocol definition. Server identifies this entry as a server definition.
:vctc.vaddr1 [TO] <i>userid vaddr2</i> [, <i>vaddr3</i> [TO] <i>userid vaddr4</i>] ...	Defines a virtual channel-to-channel device and couples it to the indicated virtual machine and address. The TO operand is optional. You can define multiple devices by specifying additional device groups (<i>vaddrn</i> [TO] <i>userid vaddrn</i> groups), with each group separated by a comma. Example: <pre>:vctc.200 to tcpip2 300, 201 tcpip2 301</pre> Applies to the stack class only.
:VMLink.vmlink-specification	VMLINK-format nicknames, minidisks, or SFS directories to be accessed. See the z/VM: CMS Commands and Utilities Reference for information about VMLINK. Example: <pre>:vmlink.* 195 tcpmaint 298 <298 G> (nonames</pre>

Table 7. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
<code>:vnic.vdev1 [TO] ownerid lanname [PORTnumber portnum] [, vdev2 [TO] ownerid lanname [PORTnumber portnum], ...]</code>	Defines a virtual network interface (NIC) at virtual device address <i>vdev1</i> and couples it to the indicated guest LAN or virtual switch (<i>ownerid lanname</i>). The TO operand is optional. You can define multiple network interfaces by specifying additional device groups (<i>vdev [TO] ownerid lanname</i> groups), with each group separated by a comma. For port based virtual switches, the PORTnumber keyword and port number can be specified. When specified, the NIC is coupled to the specified port on the virtual switch. If the network interface identified by the virtual device address does not exist, it is created and coupled to the specified guest LAN or virtual switch. If the network interface does exist, it is coupled to the specified guest LAN or virtual switch. If the port specified does not exist or cannot be used, the network interface is created but it is not coupled. An error message is displayed. If the guest LAN or virtual switch does not exist, an error message is displayed and no network interface is created. Example: <pre>:vnic.240 to system subnt240, 096 to tcpip2 subnt96</pre> Applies to the stack class only.

Customizing Servers

This section shows examples of ways to use the DTCPARMS file to customize servers.

- The following entry attaches devices 620 through 623 to the TCPIP server, and will issue a warning when file mode A is at least 90% full:

```
:Nick.TCPIP :Type.server :Class.stack
           :Attach.620-623
           :Diskwarn.90
```

- The following entry specifies that RPIVAL module will be used when validating user IDs and passwords supplied by clients for the REXECD server:

```
:Nick.REXECD :Type.server :Class.rexec
           :ESM_Validate.RPIVAL
```

- REXECD agent virtual machines can easily be defined through replicating existing servers. They only need to be described once. REXECD startup will automatically identify all of the agents to be used. The following entry identifies the RXAGENT2 VM user ID that will handle anonymous REXEC requests for the REXECD server:

```
:Nick.RXAGENT2 :type.server :class.rexec_agent
           :For.REXECD
```

- The following entry specifies for the FTPSERVE server:
 - Do not issue a warning when the A-disk is read-only.
 - The RPIVAL module will be used to validate user IDs and passwords supplied by clients.
 - The RPIUCMS command will be used for initializing and terminating a RACROUTE environment.
 - The server name, FTP server, will be displayed when the server is started.

```
:Nick.FTPSERVE :Type.server :Class.ftp
           :Diskwarn.NO
           :ESM_Validate.RPIVAL
```

```
:ESM_Racroute.RPIUCMS
:Name.FTP Server
```

Automatic Generation of Selected Startup Parameters

For certain IBM-supplied server classes, all parameters related to the use of an external security manager (ESM) or anonymous user/login support are automatically generated during the server initialization process.

The server classes, default server IDs, startup parameters, and tags/values that affect this processing are listed in [Table 8 on page 44](#). For the servers listed in this table, the parameters indicated should be omitted from any `:PARMS` tag definitions used for those servers; the tags and values shown should be used instead, to allow these parameters to be generated during server initialization.

Note: Failure to use the tags listed in [Table 8 on page 44](#) may result in incorrect or insecure operation of the identified servers.

Table 8. Server Parameters Generated at Initialization

Server Class	Default Server ID	Generated Parameter	Controlling DTCPARMS Tag/Value
rexec	REXECD	-s	:Anonymous.YES ¹
nfs	VMNFS	RN	:ESM_Enable.YES :Anonymous.YES
ftp	FTPSERVE, CSMSERVE	RACF ANONYMOU	:ESM_Enable.YES :Anonymous.YES

Note¹: For the -s parameter to be generated as a startup parameter for an REXEC server, the following conditions must be met:

- At least one DTCPARMS file entry must be present that defines a server of the `rexecd_agent` class.
- Each REXEC agent server entry must define its agent virtual machine to be a server for a particular REXEC server, through an appropriate `:For.userid` definition.
- The REXEC server entry (or the `rexec` class entry it references) must include an `:Anonymous.YES` entry.

Adding New Servers and Server Classes

Suppose you choose to add a POP server. The TCP/IP startup code does not know how to start a POP server because a POP server class is not provided by IBM. Here is an example of an entry that can be included in a DTCPARMS file to define the new server class of POPV3 and add the new POP server:

```
:Nick.POPV3           :Type.Class
:Runtime.C
:Command.POP3
:Name.Post Office Protocol Server Version 3

:Nick.POP             :Type.Server
:Class.POPV3
```

Duplicating and Running Existing Servers

With some exceptions, TCP/IP for z/VM can accommodate running more than one server of a given class with a specific TCP/IP stack server, in the event such a need arises.

Servers for which multiple server instances cannot be implemented (or, for which the steps described here are not applicable) are discussed in [Chapter 1, “Planning Considerations,” on page 1](#), [“Multiple Server Instance Restrictions” on page 9](#).

Use the following steps to duplicate an existing server that is not restricted to a single instance:

Server Duplication Steps

1. Update the CP directory to define the additional server, and define any minidisks it requires.
2. Copy the TCPROFIL EXEC file to 191 minidisk defined for the new server. Rename the copy to PROFILE EXEC.
3. Update the DTCPARMS file to identify the additional server(s).
4. Update PROFILE TCPIP, as required for your installation.

For any additional server virtual machines that you define, it is recommended that you:

- maintain any naming conventions used for that server class
- model your CP directory entries after that supplied for the virtual machine you are duplicating.

If necessary, for specific DASD storage and user ID requirements that may be applicable to the virtual machine in question, consult *Program Directory for TCP/IP for z/VM*.

For example, assume you want to define two additional FTP servers, named FTPSERV2 and FTPSERVX, where FTPSERV2 will make use of the same ports as the existing FTPSERVE server, and FTPSERVX will make use of ports 1020 and 1021. After these servers and their required resources have been defined on your system, you need to identify these servers in the DTCPARMS file. The following statements will accomplish this, specifying that the existing ftp server class be used for both servers:

```
:Nick.FTPSERV2 :type.server :class.ftp
:Nick.FTPSERVX :type.server :class.ftp :parms.port 1021
```

To allow the new servers to be managed by the TCP/IP stack, and to allow them to use the required ports, the PORT and AUTOLOG statements in the PROFILE TCPIP configuration file must be updated, as follows:

```
AUTOLOG
  FTPSERVE 0
  FTPSERV2 0
  FTPSERVX 0

PORT
  21 FTPSERVE
  21 FTPSERV2
  1021 FTPSERVX
  20 FTPSERVE NOAUTOLOG
  20 FTPSERV2 NOAUTOLOG
  1020 FTPSERVX NOAUTOLOG
```

Note that existing entries for the FTPSERVE server are maintained.

This configuration allows only the FTPSERVE and FTPSERV2 servers to listen on the well-known FTP ports 20 and 21; ports 1020 and 1021 are similarly reserved for the FTPSERVX server. Because the FTP servers only listen on port 20 and 1020 intermittently, the NOAUTOLOG operands shown prevent the stack from monitoring.

For another example, assume you want to define an LDAP server, named LDAPSRV2, in addition to the existing LDAPSRV server.

Note: Because multiple LDAP servers cannot share LDBM or GDBM databases and cannot share the LDAP server schema, each server must have a different working directory BFS file space or must have unique values for both the **databaseDirectory** configuration option (for an LDBM or file-based GDBM backend) and the **schemaPath** configuration option (for the schema). It is suggested that separate BFS file spaces be used in order to simplify deployment of additional servers. In addition, each server must listen on a different TCP port.

After the new server and its required resources have been defined on your system, you must perform these steps:

1. Identify the new server in your DTCPARMS file. **Example:**

```
:Nick.LDAPSRV2 :Type.server :Class.ldap :Parms.parms(-1 ldap://:1389)
```

This entry identifies user LDAPSRV2 as an LDAP server listening on port 1389.

2. Create a working directory BFS file space for the new LDAP server. The name of the BFS file space and the user ID of the LDAP server should be the same. Place the name in the VMSYS file pool and establish the new LDAP server as the owner of the file space. The USER option of the ENROLL USER command identifies the owner of a new BFS file space. **Example:**

```
enroll user ldapsrv2 vmsys ( bfs user ldapsrv2 blocks 2000
```

3. To allow the servers to be managed by the TCP/IP stack, and to allow them to use the required ports, update the PORT and AUTOLOG statements in the PROFILE TCPIP configuration file, as follows:

```
AUTOLOG
  LDAPSRV      0
  LDAPSRV2     0
PORT
  389          LDAPSRV
  1389         LDAPSRV2
```

Server Profile Exits

PI

Occasionally you may find that more customizing is required for a specific server than can be provided by DTCPARMS file entries. When this need arises, a server profile exit can be used. This exit is a REXX exec that you create, which receives information about a server as its initialization progresses. Based on this information, the exit may specify alternative configuration values to be used or modify some aspects of the server runtime environment.

Use the DTCPARMS file `:Exit .` tag to identify your server profile exit, and to determine the server with which it is to be used.

Note:

1. When a server profile exit and the global server exit (discussed later) are both in use, the server profile exit is called after the global exit for a given exit point.
2. Server profile exits are not applicable to GCS-based servers.
3. An exit name of TCPRUNDB should not be used as a name for an installation-defined server exit. The TCPRUNDB file name is reserved for an IBM profile exit (discussed later) that may be supplied by the IBM support team for diagnostic purposes.

Input

The information provided to the server profile exit is character data, passed as a command line argument. This data can be retrieved using either the REXX ARG(1) function call or the PARSE ARG statement.

For each exit invocation, an upper case keyword that indicates the nature (or, *type*) of the current exit point is provided. Depending on this keyword, one or more additional values may be provided. Each exit point keyword, and any applicable values, are described here:

SETUP class

The server class (as defined by a DTCPARMS file `:Class .` tag) has been identified, but no commands have been issued. The console has been spooled to the owning user ID for the server virtual machine (as determined by an applicable `:Owner .` DTCPARMS tag), or to the default owning user ID, TCPMAINT. Use the SETUP exit point to:

- Access additional minidisks needed by the server, or otherwise change the CMS search order
- Return overriding DTCPARMS values

For SETUP processing, any DTCPARMS tagged value can be changed.

BEGIN class command

The server of the indicated class will be started with the command indicated. This exit call is made after the runtime environment has been established, immediately before the server program command is issued. For BEGIN processing, only these DTCPARMS tagged values can be modified:

```
:Command :Parms :Exit
```

END class return_code command

The server program has ended with the indicated return code.

ADMIN

A configuration problem that is under system administrative control has been encountered. An explanatory error message has been displayed on the server console.

ERROR return_code command

The server cannot be started due to the failure of a needed CP or CMS command. The failing command and its return code are passed as additional parameters. An error message has been displayed on the server console.

Output

Server profile exit return codes must be specified as the first token of any expression specified for a REXX RETURN or EXIT statement. If no return code is given, the server initialization program assumes a return code of zero (0).

Return Codes

For SETUP and BEGIN exit calls, the following return codes have meaning:

0

Server initialization is to continue.

4

Server initialization is to be cancelled. The server virtual machine is to remain logged on.

other

For all other return codes, server initialization is cancelled, and the server virtual machine is logged off if it is running disconnected.

Return codes provided upon completion for other types of exit calls are ignored by the initialization program.

DTCPARMS Override Values

When the server exit returns control with a return code of zero (either directly, or by default) overrides to DTCPARMS file values are accepted by the initialization program for these exit points:

SETUP

any DTCPARMS tagged value can be changed.

BEGIN

Only these DTCPARMS tagged values can be modified:

```
:Command :Parms :Exit
```

Note: DTCPARMS overrides provided upon completion for other types of processing are ignored by the initialization program.

Examples

Because server profile exits are intended to accommodate unique installation requirements, a sample exit of this type is not provided as part of TCP/IP for z/VM. However, the sample *global profile exit* (discussed in the next section) can be used as a guide for developing such an exit, since it utilizes the same input and output conventions just described.

The examples that follow illustrate different methods of returning DTCPARMS override values.

In this example, assume the server profile exit incorporates logic for accommodating certain External Security Manager (ESM) requirements. After having been called for SETUP processing, the exit determines appropriate DTCPARMS values to return, for which this statement is used:

```
RETURN ":ESM_Enable.YES :ESM_Validate.VALIDATE"
```

Here, no return code has been specified as part of the RETURN expression, so the initialization program will assume a return code of zero. The server will be started using the indicated DTCPARMS file values — in place of any previously defined :ESM_Enable. and :ESM_Validate. values, but in addition to those defined by other DTCPARMS file tags.

For this next example, assume the server profile exit is used to facilitate testing of a newly acquired (or modified) MODULE file, for use by the FTP server machine. For testing purposes the module is maintained with a unique name, as is its corresponding configuration file. For a BEGIN processing call, the exit might use this RETURN statement :

```
RETURN "0:Command.SRVFTPX1 :Parms.SRVFTPX1 TESTCFG *"
```

In this case, the return code of 0 is specified as the first token of the RETURN expression. Server startup will continue, using the specified command and parameters as overrides to any previously defined :Command. and :Parms. values.

Global Profile Exit

To accommodate processing requirements that are common to all servers or a group of similar servers, the *global profile exit* (TCPRUNXT EXEC) can be used. This global server exit utilizes the same inputs and outputs as those defined for locally developed server profile exits.

The TCPRUNXT EXEC, if it exists, is automatically invoked by the server initialization program; this exit does not need to be identified within a DTCPARMS file.

Note:

1. When both a server profile exit and the global server exit are in use, the server profile exit is called after the global exit for a given exit point.
2. The global profile exit is not applicable to GCS-based servers.

Within the global profile exit, the REXX USERID() function — in addition to the *class* argument passed to the exit — may prove useful for identifying when a specific server is being initialized.

Global Profile Exit Sample

A sample global profile exit is provided as TCPRUNXT SAMPEXEC on the TCPMAINT 591 minidisk. Your customized exit should be maintained on the TCPMAINT 198 minidisk, with a file type of EXEC. For more information about the supplied global server exit, review the content of the TCPRUNXT SAMPEXEC file.

IBM Diagnostic Profile Exit

At times, assistance from the IBM support center may be required to resolve problems that are associated with a specific protocol server. For such circumstances, it might be necessary to obtain detailed problem information beyond what can be acquired through existing means (such as conventional messages or data files, including those produced by using server-specific tracing facilities). For example, to diagnose certain problems, a virtual machine dump based on a specific trap condition might be required, or the virtual machine operational environment might need to be examined in detail. To accommodate processing requirements that are unique to gathering such information, an IBM diagnostic profile exit (TCPRUNDB EXEC) can be used. This diagnostic exit utilizes the same inputs and outputs as those defined for locally developed server profile exits. Thus, it can be used in conjunction with those exits, without requiring changes to them in order to facilitate problem diagnostic operations.

The TCPRUNDB EXEC, if it exists, is automatically invoked by the server initialization program; this exit does not need to be identified within a DTCPARMS file.

Note:

1. When the diagnostic profile exit is present, it is called after any global or server profile exit for a given exit point.

2. The IBM diagnostic profile exit is not applicable to GCS-based servers.
3. Because the IBM diagnostic profile exit is intended for use only when problems are being investigated in conjunction with the IBM support team, a TCPRUNDB exit (executable or otherwise) is not provided as part of TCP/IP for z/VM. Instead, a TCPRUNDB EXEC customized to aid in the diagnosis of a specific problem, may be provided by the IBM support team, as dictated by problem circumstances.

Customizing Server-specific Exits

In addition to the server profile exits just described, several server-specific exits are provided (in sample form) that can be used with certain TCP/IP servers. In general, the exits listed in [Table 9 on page 49](#) allow you to customize various operational aspects for a given server, to better control how its services are provided by your installation, or to identify a set of users who are authorized to manage a particular server.

Table 9. TCP/IP Server-specific Exits

TCP/IP Server	Supported Exits		
TCP/IP Stack	SCEXIT	PMEXIT	
FTP	FTPEXIT	CHKIPADR	
NFS	VMNFSCMS	VMNFSSMON	VMNFSSMG
SMTP	SMTPCMDX	SMTPFWDX	SMTPVERX
UFT	FTPEXIT	CHKIPADR	

When the server exits listed in [Table 9 on page 49](#) are customized to meet the needs of your installation, note that commands which make use of TCP/IP services should **not** be used within any of those exits. Examples of programs and functions that should not be utilized are:

- PING, NSLOOKUP, REXEC
- TCP/IP- oriented CMS PIPE stages and Rexx Sockets APIs
- Local or third-party applications that use TCP/IP socket interfaces or other services

For the TCP/IP stack server, doing so creates a situation where the stack is required to provide TCP/IP services to itself, at which point internal interrupt handling and blocking issues then arise. For other TCP/IP servers, similar problems can result, not just with VMCF or IUCV communication interrupt handling, but also with TCP/IP connection management.

Attempts to make use of a secondary TCP/IP "worker" stack to provide the desired information will also encounter the same problems just described, regardless of the servers involved.

Care must be taken to ensure that all commands or programs invoked within a TCP/IP server exit do not adversely affect the operation of that server. This includes not just TCP/IP-oriented commands, but those which can cause CMS storage management changes, extended wait conditions, or that otherwise adversely affect server performance. Without such care, unpredictable results or other operational errors can and may occur.

When more than one virtual machine is used to provide support for a given TCP/IP service, it might be necessary to incorporate appropriate logic within a given exit program to account for its use by multiple servers. For example, the FTP server CHKIPADR exit, because it cannot be copied and renamed for use in multiple FTP servers, might require specific modifications to ensure satisfactory operation if it is used by multiple FTP servers.

GCS Servers

GCS servers use a *userid* GCS file, and if customization is needed, you should modify this file on TCPMAINT 198 at installation. GCS servers, like CMS servers, will display a prompt if the console is connected. The reply is "REPLY *nn*" to start or "REPLY *nn* X" to cancel.

PI end

TCP/IP Configuration File Overview

This section presents a table that lists the various TCP/IP configuration files that may be referenced by either TCP/IP clients, a specific TCP/IP server, or both. For each configuration file, Table 10 on page 50 lists each (production) configuration file, its sample counterpart (if one is supplied by IBM), the minidisk where the *sample* file resides, and a reference to the chapter that provides details about the content and use of that file.

The first five files listed in Table 10 on page 50 are necessary to provide basic TCP/IP services for most environments. The first file, IBM DTCPARMS, contains server configuration definitions. The next three files, PROFILE TCPIP, HOSTS LOCAL, and ETC HOSTS, are configuration files for the TCPIP server virtual machine. The next two files, TCPIP DATA and ETC SERVICES, need to be accessible to all TCP/IP servers, applications, and users; these files contain information that is (or may be) referenced by all users. The remaining files are for various server virtual machines that you might have installed. Most sites will need to create the PROFILE TCPIP, HOSTS LOCAL or ETC HOSTS, and TCPIP DATA configuration files.

Table 10. Configuration Files and Minidisk Location Summary

Production Configuration File	IBM-Supplied Sample File	Disk	Reference Location
IBM DTCPARMS	none	591	Chapter 5, “General TCP/IP Server Configuration,” on page 33
PROFILE TCPIP	PROFILE STCPIP	591	Chapter 16, “Configuring the TCP/IP Server,” on page 507
HOSTS LOCAL	HOSTS SLOCAL	592	Chapter 4, “Configuring the Local Host Files,” on page 27
ETC HOSTS	ETCHOSTS SAMPLE	592	Chapter 4, “Configuring the Local Host Files,” on page 27
TCPIP DATA	TCPIP SDATA	592	Chapter 3, “Defining the TCP/IP System Parameters,” on page 13
ETC SERVICES	ETC SAMPSEV	592	None; refer to comments within the supplied file.
MPROUTE CONFIG	MPROUTE SCONFIG	591	Chapter 8, “Configuring the MPRouter Server,” on page 193
SRVRFTP CONFIG	SRVRFTP SCONFIG	591	Chapter 6, “Configuring the FTP Server,” on page 55
CSMSERVE CONFIG	CSMSERVE SCONFIG	591	Chapter 6, “Configuring the FTP Server,” on page 55
RSCSTCP CONFIG	RSCSTCP SCONFIG	591	Chapter 12, “Configuring the RSCS Print Server,” on page 353
RSCSLPD CONFIG	RSCSLPD SCONFIG	591	Chapter 12, “Configuring the RSCS Print Server,” on page 353
RSCSLPR CONFIG	RSCSLPR SCONFIG	591	Chapter 12, “Configuring the RSCS Print Server,” on page 353
RSCSLPRP CONFIG	RSCSLPRP SCONFIG	591	Chapter 12, “Configuring the RSCS Print Server,” on page 353
SMTP CONFIG	SMTP SCONFIG	591	Chapter 13, “Configuring the SMTP Server,” on page 375
SMTP RULES	none	-	Chapter 13, “Configuring the SMTP Server,” on page 375
UFTD CONFIG	UFTD SCONFIG	591	Chapter 17, “Configuring the UFT Server,” on page 649
VMNFS CONFIG	VMNFS SCONFIG	591	Chapter 9, “Configuring the NFS Server,” on page 325

Table 10. Configuration Files and Minidisk Location Summary (continued)

Production Configuration File	IBM-Supplied Sample File	Disk	Reference Location
PW SRC	none	-	Chapter 14, “Configuring the SNMP Servers,” on page 439
SNMPTRAP DEST	none	-	Chapter 14, “Configuring the SNMP Servers,” on page 439

If you need to customize a specific configuration file, note the following:

- IBM-supplied end-user sample files are supplied on the TCPMAINT 592 minidisk. When such a file is customized, the *sample* file should be copied to this same disk (TCPMAINT 592) as its *production* file name and type; changes should then be made to the *production* configuration file.
- IBM-supplied server-related sample files are supplied on the TCPMAINT 591 minidisk. When such a file is customized, the *sample* file should be copied to the *configuration* disk (TCPMAINT 198) as its *production* file name and type; changes should then be made to the file on the configuration disk.

Note:

1. For optional TCP/IP services, you need to configure only those files referenced by the TCP/IP servers you plan to use.
2. Because the PW SRC file contains passwords, you should control access to these files if security is a concern.

Server Administrative Command Interface Summary

The services provided or managed by a specific TCP/IP server virtual machine can often be stopped (and to a lesser extent, started or altered) by using a server-specific interface that provides support for dynamic operations. The TCP/IP servers that support dynamic operations in some manner are listed here:

- TCP/IP stack, through the NETSTAT and OBEYFILE commands
- MPRoute server, through an SMSG command interface
- FTP server, through an SMSG command interface
- LDAP server, through an SMSG command interface
- NFS server, through an SMSG command interface
- SMTP server, through an SMSG command interface
- SSL server, through its associated SSLADMIN command interface
- UFT server, through console commands.

For detailed information about dynamic operational support for these servers, refer to the respective server configuration chapters.

Stopping TCP/IP Servers

When it is necessary to stop a specific TCP/IP server for which there is no dynamic control interface (or no "stop" or "shutdown" command is available), use the following procedure:

1. Log on to the server to be stopped.
2. Enter the `#CP EXTERNAL` command to initiate the shutdown of the server. In some cases, other prompts may be issued to determine if the shutdown of the service machine should continue; the text of these prompts varies somewhat from server to server. Also, you should allow sufficient time for the shutdown to complete before issuing additional commands.

If you receive the message:

```
DMSHDE744R Unexpected external interrupt detected,
interrupt status consists of:
```

```
CODE=0040, CPUID=0000, PARAMETER=00000000.  
Enter a 1 for ABEND or a 2 for RESUME:
```

enter 1. You will return to the CMS command line.

You can now manipulate any files that may have been created while the server was running.

Note:

1. You can use the NETSTAT CP EXTERNAL command to issue an interruption to the TCP/IP stack. Issuing this command effectively shuts down TCP/IP and any additional servers it controls. For more information on the NETSTAT CP command, see [NETSTAT Command in z/VM: TCP/IP User's Guide](#).
2. The #CP EXTERNAL command should not be used to stop the MPRoute or SSL server.
 - The command to stop MPRoute is #CP SMSG * SHUTDOWN. For information on the MPRoute server, see [Chapter 8, "Configuring the MPRoute Server," on page 193](#).
 - The command to stop the SSL server is SSLADMIN STOP. For more information, see ["SSLADMIN STOP Command" on page 498](#).
3. The shutdown of an SSL DCSS Management Agent server (SSLDCSSM, or its equivalent) is managed by the TCP/IP server initialization program (TCPRUN). A server of this type is stopped at the same time as its associated TCP/IP stack server, when the latter is configured to support secure communications. While an SSL DCSS Management Agent server can be stopped through external means, it is recommended that the startup of such agent servers remain under TCPRUN control.

Starting TCP/IP Servers

The AUTOLOG1 virtual machine, which is automatically logged on during z/VM initialization, can be used to automatically start the TCP/IP server through its PROFILE EXEC. The PROFILE EXEC provided by z/VM on AUTOLOG1's 191 disk contains an XAUTOLOG command for the default TCP/IP server, TCPIP, which is commented out. By uncommenting this XAUTOLOG command for TCPIP in AUTOLOG1's PROFILE EXEC, the TCPIP server is automatically logged on when z/VM is IPLed. See "Steps for automatically starting TCP/IP" in [z/VM: Getting Started with Linux on IBM Z](#) for information on uncommenting this XAUTOLOG command for TCPIP in AUTOLOG1's PROFILE EXEC to have the TCPIP server automatically started when z/VM is IPLed.

In most cases, individual TCP/IP servers can be started as needed by using the CP XAUTOLOG command, or by logging on to a server and executing its PROFILE exec and then providing appropriate responses to any prompts that are issued. Starting servers in this way may prove useful when initially configuring TCP/IP services or diagnosing problems.

However, it is recommended that the TCP/IP server be allowed to manage initialization of an SSL server pool, through use of the SSLADMIN START or NETSTAT SSL START commands. The use of another mechanism for this purpose (individual XAUTOLOG commands, or use of the AUTOLOG1 virtual machine) could allow conditions to arise that can inhibit successful startup of such servers, which then might impact the ability for secured connections to be established as required for your installation.

Also, note that startup of an SSL DCSS Management Agent server (SSLDCSSM, or its equivalent) is managed by the TCP/IP server initialization program (TCPRUN). A server of this type is initialized at the same time as its associated TCP/IP stack server, when the latter is configured to support secure communications. While an SSL DCSS Management Agent server can be started through external means, it is recommended that the startup of such agent servers remain under TCPRUN control.

TCP/IP and SSL Server Logon Restrictions

A Telnet connection established with a z/VM host cannot be used to logon the TCP/IP server virtual machine that manages that same connection. An attempt to do so will be rejected, with this message reported:

HCPLGA206E

Cannot connect to host virtual machine

If logon of the TCP/IP server is necessary, this should be accomplished through use of a Telnet connection managed by a different TCP/IP server, or via some other communication path.

Similarly, a secure Telnet connection should not be used to logon an SSL server using a communication path that involves execution of that same SSL server. A logon attempt of this type will cause the subject Telnet connection, as well as the SSL server and any secure connections it manages, to hang. This situation is possible because the logical devices for all Telnet sessions (secure or unsecure) are created by the TCP/IP server, which precludes the ability of CP to detect and prevent such a logon attempt.

If logon of the SSL server is necessary, this should be accomplished through use of an unsecure Telnet connection, a secure connection provided by an SSL-TCP/IP server pairing different from those associated with the SSL server of interest, or via some other communication path.

Chapter 6. Configuring the FTP Server

The File Transfer Protocol (FTP) virtual machine serves client requests to transfer files between TCP/IP hosts to or from your VM host. To configure the FTP server, you must perform the following steps:

FTP Server Configuration Steps
1. Update the TCP/IP server configuration file. 2. Update the DTCPARMS file for the FTP server. 3. Establish FTP server machine authorizations. 4. Customize the FTP server configuration file. 5. Configure automatic file translation. (optional) 6. Configure secure FTP connections. (optional) 7. Customize FTP server exits. (optional)

Dynamic Server Operation: The FTP server provides a VM Special Message (MSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information, see [“MSG Interface to the FTP Server”](#) on page 74.

Step 1: Update PROFILE TCPIP

Include the FTP server virtual machine in the AUTOLOG statement of the TCPIP server configuration file. The FTP server is then started automatically when TCPIP is initialized. The IBM default user ID for this server is FTPSERVE. Verify that the following statements have been added to PROFILE TCPIP:

```
AUTOLOG
  FTPSERVE 0
```

The FTP server requires that ports TCP 20 and TCP 21 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
  20 TCP FTPSERVE NOAUTOLOG ; FTP Server
  21 TCP FTPSERVE           ; FTP Server
```

Step 2: Update the DTCPARMS File

When the FTP server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the FTP server are:

```
:Nick.FTPSERVE
:Anonymous.
:ESM_Enable.
:ESM_Validate.
:ESM_Racroute.
:Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Note: You should modify the DTCPARMS file for the FTP server if you:

- Use a configuration file other than SRVRFTP CONFIG, such as CSMSEVERE CONFIG (see the “TCP/IP configuration changes” section in the [z/VM: Service Guide](#) for more information)

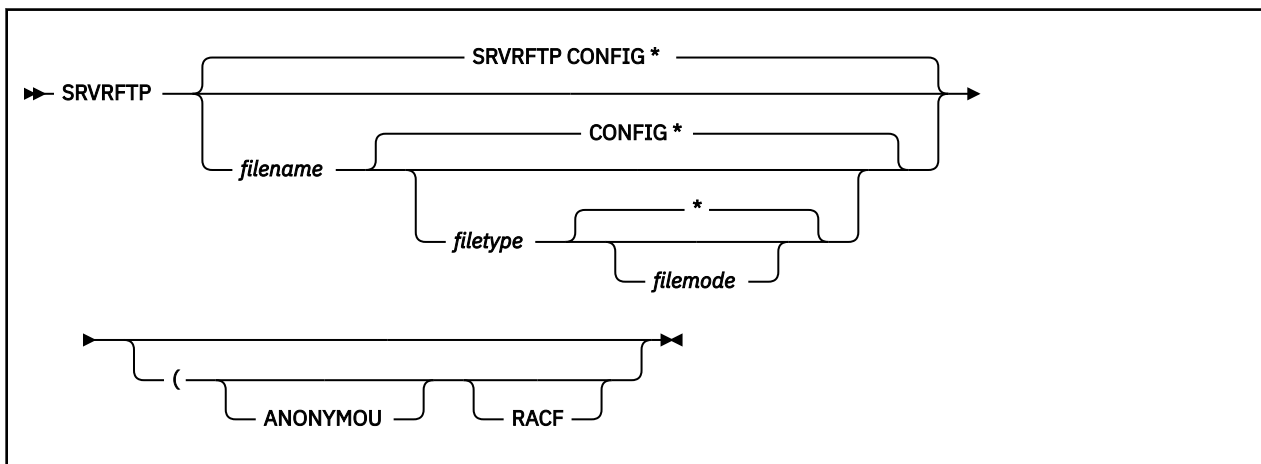
Configuring the FTP Server

- Provide anonymous FTP support
- Use an ESM for client authorization and access control.

SRVRFTP Command Operands (:Parms. Parameters)

FTP services are initiated using the SRVRFTP command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with an FTP server definition. For more information about this command and its operands, see [“SRVRFTP Command Syntax” on page 56.](#)

SRVRFTP Command Syntax



Operands

filename

The file name of the FTP server configuration file. The default file name is SRVRFTP.

filetype

The file type of the configuration file. The default file type is CONFIG.

filemode

The file mode of the configuration file. The default file mode is an asterisk (*).

ANONYMOU

Directs the FTP server to allow a client to login with a user name (ID) of either "anonymous" or "anonymou" without requiring a logon password. This operand is automatically supplied when an Anonymous . YES entry is specified in the DTCPARMS file.

RACF

Directs the FTP server to rely upon an External Security Manager (ESM) to control access to minidisks. This operand is automatically supplied when an ESM_Enable . YES entry is specified in the DTCPARMS file. For information on the FTP server's interaction with the ESM, see [Appendix A, “Using TCP/IP with an External Security Manager,” on page 685.](#)

Step 3: Establish FTP Server Machine Authorizations

In order for FTP clients to access files or directories in the CMS Shared File System (SFS), the FTP server must have SFS file pool administrator authority. Each FTP server that will provide such access must be listed on the ADMIN statement in the SFS file pool server's DMSPARMS file. For details on SFS file pool configuration and administrator authority, see [z/VM: CMS File Pool Planning, Administration, and Operation.](#)

In order for FTP clients to access files and directories in the Byte File System (BFS), the FTP server must be defined as a POSIX "superuser". To allow this capability, the following statement must be included in the CP directory:

```
POSIXINFO UID 0 GID 0
```

See *z/VM: OpenExtensions User's Guide* and *z/VM: CP Planning and Administration* for more information about configuring the FTP server in this manner.

The CP directory entry for the FTP server must include an **OPTION DIAG 88** statement and the server must have **class B** privileges, regardless of whether an External Security Manager (ESM) is in use.

If FTP virtual reader support is enabled, the FTP server virtual machine must also have **class D** privileges.

The FTP server can use an external security manager (ESM) to authenticate FTP clients and to control access to z/VM resources. To use an ESM, specify :ESM_Enable.Yes in the DTCPARMS file. For more information, see Appendix A, "Using TCP/IP with an External Security Manager," on page 685.

Step 4: Customize the FTP Server Configuration File

There are two distinct FTP servers supplied with z/VM — the FTPSERVE server, intended for general FTP support and use, and the CSMERVE server, intended for dedicated use by z/VM Centralized Service Management (z/VM CSM). Different sample configuration files are supplied for each server, and should be used as the basis for each of the configuration files discussed here.

SRVRFTP CONFIG

This FTP configuration file defines how the general-use FTP server is to operate and which services and types of access it provides. A sample general-use FTP configuration file is shipped with function level 730 as SRVRFTP SCONFIG on the TCPMAINT 591 disk. Your customized SRVRFTP CONFIG file should be stored on the TCPMAINT 198 minidisk.

CSMERVE CONFIG

This dedicated FTP server configuration file defines how the FTP server for z/VM Centralized Service Management (z/VM CSM) is to operate and which services and types of access it provides. A sample dedicated FTP server configuration file is shipped with function level 730 as CSMERVE SCONFIG on the TCPMAINT 591 disk. Your customized CSMERVE CONFIG file should be stored on the TCPMAINT 198 minidisk.

Specifying entries within the configuration file

Within the configuration file, blanks and record boundaries are used to delimit tokens. All characters to the right of, and including, a semicolon (;) are treated as a comment.

FTP Server Configuration File Statements

ANONYMOU Statement

Purpose

The ANONYMOU statement directs the FTP server to allow a client to login with a user name (ID) of either "anonymous" or "anonymou" without requiring a logon password.

Note: The recommended method for enabling anonymous FTP support is to specify an :Anonymous.YES entry in the DTCPARMS file instead of using the ANONYMOU configuration statement.

```
➤ ANONYMOU ➤
```

Operands

The ANONYMOU statement has no operands.

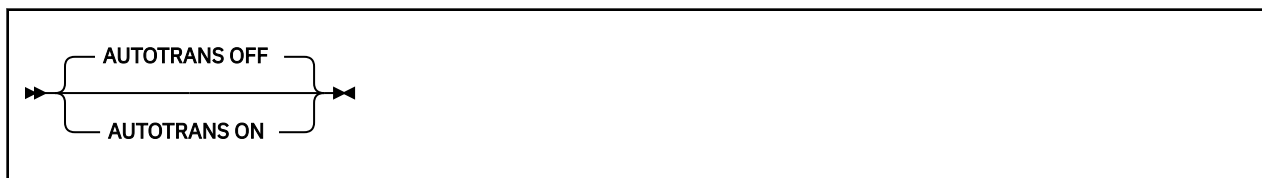
Usage Notes

- For installations that make use of External Security Manager (ESM) (those for which an :ESM_Enable . YES entry has been specified in the DTCPARMS file, and which may make use of the FTP server RACF configuration statement), the user ID ANONYMOU must be defined to the ESM that is in use.
- The user ID ANONYMOU must be enrolled in any SFS file pool for which anonymous access is to be allowed.
- It is not necessary for the ANONYMOU user ID to be defined in the CP directory when anonymous FTP support is enabled.

AUTOTRANS Statement

Purpose

The AUTOTRANS statement specifies whether automatic file translation is performed by default when files are transferred using the **Image** transfer type.



Operands

ON

Specifies that automatic file translation should be enabled as the default when an FTP session is established.

OFF

Specifies that automatic file translation should be disabled as the default when an FTP session is established.

The specified default translation setting is applied to all Image transfers that are requested by clients unless:

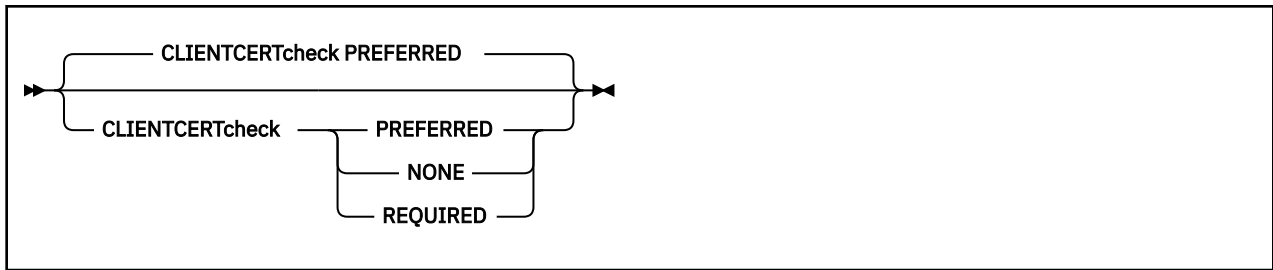
- The CHKIPADR exit has been configured to select a specific translation setting when a user logs in
- The automatic translation setting is changed during an FTP session by a client via the AUTOTRANS operand of the SITE subcommand.

Selection of the translation table to be used for automatic file translation is discussed in [Chapter 19, “Using Translation Tables,”](#) on [page 667](#). An FTP client can also explicitly specify the translation table to use on the server with the SITE TRANSLATE command. See [z/VM: TCP/IP User's Guide](#) for more information about the SITE TRANSLATE command.

CLIENTCERTCHECK Statement

Purpose

The CLIENTCERTCHECK statement specifies the level of client certificate checking.



Operands

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

DONTREDIRECT Statement

Purpose

The `DONTREDIRECT` statement restricts an FTP client to establishing data connections with only the local system. This prevents a client from using the FTP server to mount an "FTP bounce" attack, but violates the File Transfer Protocol RFC (RFC 959) which allows a data connection to be established between two remote systems.

No applications that exploit this capability are known to exist. However, if you activate this option, you should be aware of the potential limitation it imposes on legitimate applications of FTP.

```
➡ DONTREDIRECT ➡
```

Operands

The `DONTREDIRECT` statement has no operands.

FTAUDIT Statement

Purpose

The `FTAUDIT` statement causes the FTP server exit (FTPEXIT) to be loaded during initialization and enables audit processing.

For audit processing, this exit is called for every transfer of data (bytes) over a connection; this includes data associated with files, as well as data returned in response to `LIST`, `DIR` or similar subcommands. Events such as login and logout (that is, the `USER` and `QUIT` subcommands) will also be audited.

```
➡ FTAUDIT ➡
```

Operands

The FTAUDIT statement has no operands.

FTCHKCMD Statement

Purpose

The FTCHKCMD statement causes the FTP server exit (FTPEXIT) to be loaded during initialization and enables general command exit processing.

The FTP exit is called to perform command validation for every received FTP subcommand. The server command exit can be used to perform additional validation of a supplied user ID, IP address, or the subcommand itself. If appropriate, the exit can then indicate the supplied subcommand should be rejected, with an exit-defined message returned to the user.

➤ FTCHKCMD ➤

Operands

The FTCHKCMD statement has no operands.

FTCHKDIR Statement

Purpose

The FTCHKDIR statement causes the FTP server exit (FTPEXIT) to be loaded during initialization and enables CD command exit processing.

The exit is called to validate FTP directory changes, allowing greater control over access to system resources by providing the capability to selectively honor or refuse a client CD (Change Directory) request.

➤ FTCHKDIR ➤

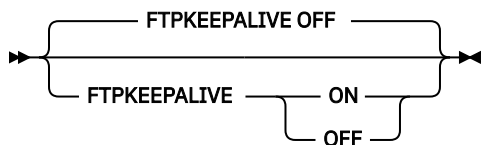
Operands

The FTCHKDIR statement has no operands.

FTPKEEPALIVE Statement

Purpose

The FTPKEEPALIVE statement defines if the FTP server will use TCP/IP stack's keepalive timer value for control connections.



Operands

ON

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

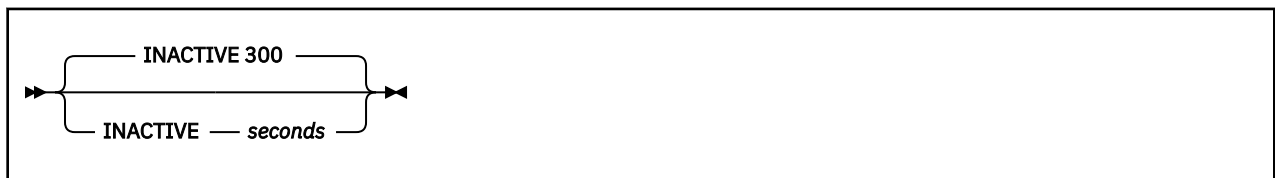
OFF

Specifies that the FTP server should allow idle control connections to time out.

INACTIVE Statement

Purpose

The `INACTIVE` statement sets the inactivity time-out the FTP server should apply to connections, once they are established. The FTP server closes connections found to be inactive for the specified amount of time.



Operands

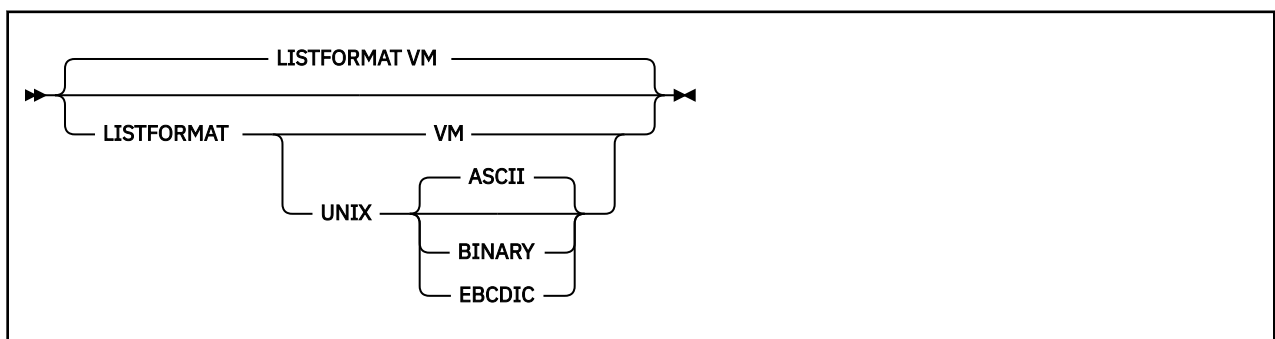
seconds

The number of seconds of inactivity after which the FTP server will close a connection. Specify *seconds* as an integer between 1 and 1,048,576. The default inactivity time-out is 300 seconds (5 minutes).

LISTFORMAT Statement

Purpose

The `LISTFORMAT` statement sets the format default for list information supplied by the FTP server when it responds to client `DIR` (or, `LIST`) requests. If this statement is not specified, the default format is **VM**.



Operands

VM

Specifies that VM-format lists should be supplied by default when responding to client `DIR` (or, `LIST`) requests.

UNIX

Specifies that UNIX-format lists should be supplied by default when responding to client `DIR` (or, `LIST`) requests. Using `LISTFORMAT` with `UNIX`, you can specify the transfer mode (`ASCII`, `BINARY`,

or EBCDIC) whose resulting size should be shown as file size in the output of the DIR (or, LIST) subcommands. The default transfer mode is ASCII.

The specified format default is applied to all LIST responses supplied to clients unless:

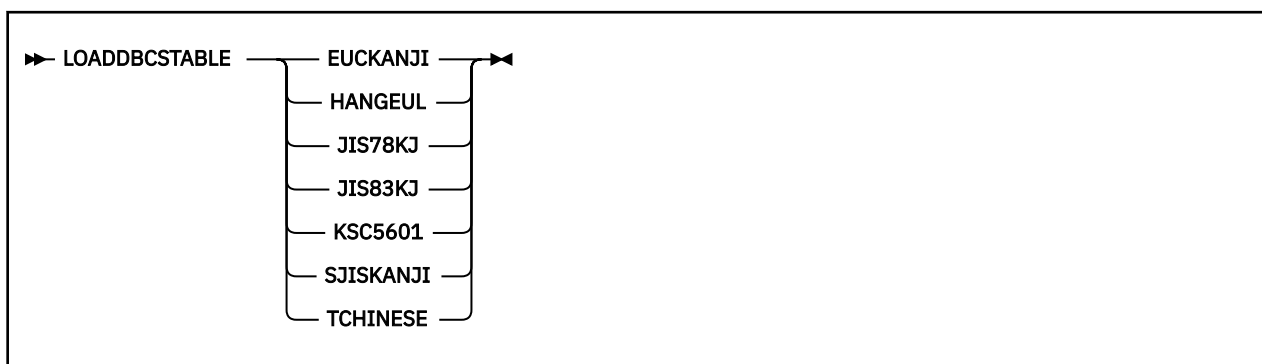
- The CHKIPADR exit has been configured to select a specific format default when a user logs in
- the list format is changed during an FTP session by a client via the LISTFORMAT operand of the SITE subcommand.

For detailed information about VM-format and Unix-format responses, see the [z/VM: TCP/IP User's Guide](#).

LOADDBCSTABLE Statement

Purpose

The LOADDBCSTABLE statement identifies DBCS translate tables that are to be loaded when the FTP server is initialized. By using multiple LOADDBCSTABLE statements, any number of translate tables may be selected ranging from none to all supported DBCS translate tables.



Operands

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.

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.

KSC5601

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

SJISKANJI

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

TCHINESE

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

Usage Notes

- Additional virtual storage may need to be defined for the FTP server if a large number of translate tables are loaded concurrently.
- The IBMKANJI transfer type does not require any translate table to be loaded. For more information on loading and customizing DBCS translate tables, see [“Customizing DBCS Translation Tables”](#) on page 673.

PASSIVEPORTRANGE Statement

Purpose

The PASSIVEPORTRANGE statement restricts the ports used by the FTP server during passive data transfer. Stateful firewalls normally watch for PASV and PORT verbs on the control connection. However, this is not possible when traffic is encrypted. The passive port range allows administrators to know which ports the FTP client may select for FTP clients to connect to for passive data transfer.

```
➤➤ PASSIVEPORTRANGE — low_port — high_port ➤➤
```

Operands

low_port

Specifies the starting port number of the port range to restrict passive data transfer to.

high_port

Specifies the ending port number of the port range to restrict passive data transfer to.

PORT Statement

Purpose

The PORT statement causes the FTP server to listen on a specified TCP connection port. By convention, port number 21 is usually reserved (in the TCPIP server configuration file) for the FTP server to accept FTP connection requests.

```
➤➤ { PORT 21 } ➤➤
   { PORT — port_number } ➤➤
```

Operands

port_number

An integer in the range of 1 through 65,534 that specifies the port number to which the FTP server listens. The default is port 21.

RACF Statement

Purpose

The RACF statement directs the FTP server to rely upon an External Security Manager (ESM) to control access to minidisks.

Note: The recommended method for enabling the use of an ESM is to specify an `:ESM_Enable.YES` entry in the DTCPARMS file instead of using the RACF configuration statement.

For more information about using an external security manager, see [Appendix A, “Using TCP/IP with an External Security Manager,”](#) on page 685.

➤ RACF ➤

The RACF statement has no operands.

Usage Notes

When this statement is used, an `:ESM_Enable.YES` entry must also be included in the DTCPARMS file to allow ESM-specific initialization processing to be performed.

RDR Statement

Purpose

The RDR statement enables FTP server reader file support, which allows files to be transferred to a VM user's virtual reader.

➤ RDR — *filemode* ➤

Operands

filemode

The file mode of the resource to be used to store files temporarily before they are sent to a virtual reader. Any of the following resources may be used for this purpose:

- a minidisk
- a temporary minidisk
- a virtual disk
- an SFS directory.

Usage Notes

- When FTP reader file support is enabled, users are allowed to STOR files to a virtual reader of a VM user ID. To allow users to issue DELETE and DIR/LS commands against a reader directory, the FTP server virtual machine must have class D privileges. (Class D is required for the CP PURGE *userid* RDR *spoolid* and CP QUERY RDR *userid* commands.)
- The FTP reader file support may be disabled by the FTP general command exit or the CD command exit.
- An SFS directory cannot be used for both temporary RDR file storage and as a substitute for the FTP server "A" disk. The FTP server requires a minidisk to be accessed at file mode A for proper operation.

For example, assume you choose to use the FTP server "root" SFS directory in file pool MYFP00L1 as the storage area for files directed to a user's virtual reader. The following DTCPARMS entry (added to the appropriate FTP *server* or *class* definition) will configure the FTP server to acquire this resource with file mode F:

```
:nick.FTPSERVE  
:vmlink. .DIR MYFP00L1:FTPSERVE. <★ F>
```

Additionally, a corresponding **RDR F** statement must be included in the FTP server configuration file. File mode F is used here to ensure the MYFP00L1:FTPSERVE. directory is accessed in the CMS search order after the TCP/IP configuration, client-code, and server-code minidisks.

For installations that rely on external security manager (ESM) services, the use of a minidisk or virtual disk is the preferred method for accommodating FTP server RDR support. If you do decide to use an SFS directory as the storage area for FTP server RDR support with an external security manager, each FTP login user that puts files to a reader may need to be given read authority to that SFS directory in order for a PUT to succeed.

- Once a file is transferred to the designated temporary RDR resource, the file is written to the client-specified reader directory. During this latter process, the FTP service machine cannot process any other FTP connection requests. For this reason, FTP client connections may time out when very large files are transferred to a reader directory.

RESTRICTUSE Statement

Purpose

The **RESTRICTUSE** statement, when processed, restricts use of the dedicated FTP server to the specified user ID only. When this statement is processed by the FTP server module (SRVRFTP), it will cause the dedicated FTP server to accept LOGIN requests from the specified user ID only. In addition, this statement prevents certain other statements (if present) from being honored.

This statement should be included in only the CSMERVE CONFIG file that is used by the CSMERVE dedicated FTP server.

►► **RESTRICTUSE** — *user_ID* ◄◄

Operands

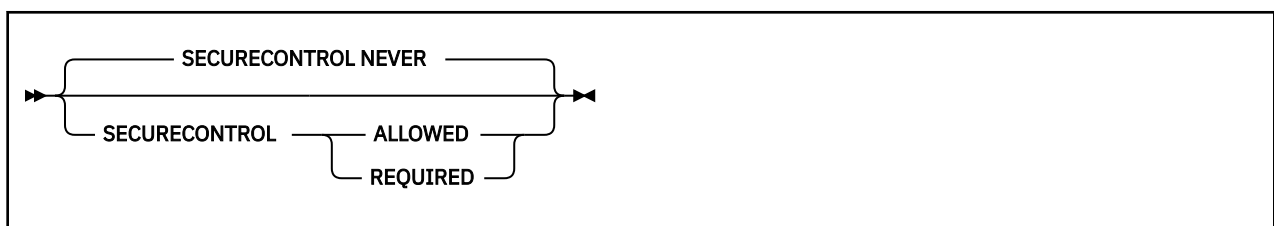
user_ID

The user ID to which use of the dedicated FTP server is restricted.

SECURECONTROL Statement

Purpose

The SECURECONTROL statement specifies the FTP server-wide minimum security level for control connections.



Operands

ALLOWED

Secure control connections using TLS are optional. When `ALLOWED` is specified, the FTP client may optionally secure the control connection using TLS. Otherwise, the control connection will not be secured using TLS.

REQUIRED

Secure control connections using TLS are required. When REQUIRED is specified, clear control connections are not allowed.

NEVER

Secure control connections using TLS are not allowed. This is the default.

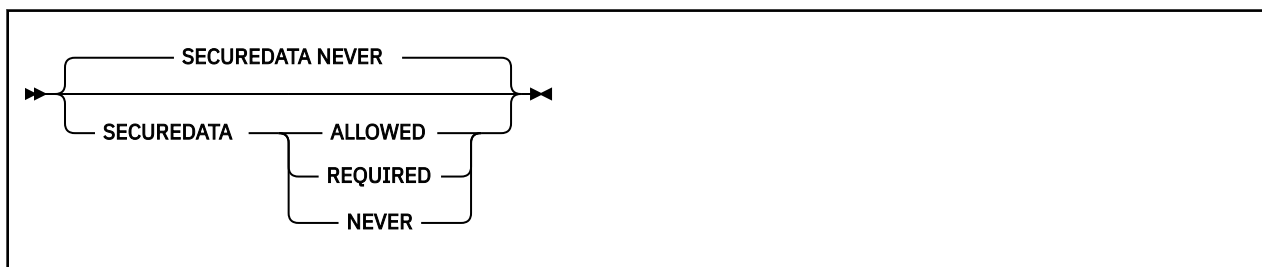
Usage Notes

- The secure control connection setting may be set dynamically using the FTP server `MSG SECURE` command.
- The `SECURECONTROL` statement does not affect statically secured control connections.

SECUREDATA Statement

Purpose

The `SECUREDATA` statement specifies the FTP server-wide minimum security level for data connections.



Operands

ALLOWED

Secure data connections using TLS are optional. When `ALLOWED` is specified, the FTP client may optionally secure data connections using TLS. Otherwise, data connections will not be secured using TLS.

REQUIRED

Secure data connections using TLS are required. When `REQUIRED` is specified, clear data connections are not allowed.

NEVER

Secure data connections using TLS are not allowed. This is the default.

Usage Notes

- The secure data connection setting may be set dynamically using the FTP server `MSG SECURE` command.
- The `SECUREDATA` statement does not affect statically secured data connections.

SYSTEMGREETING Statement

Purpose

The `SYSTEMGREETING` statement specifies whether the initial system greeting line (containing the FTP server level and domain name information) is given to FTP clients when they connect to the FTP server. System administrators may want to suppress this information for security reasons.



Operands

ON

Specifies that the FTP server level and domain name information will be given in the FTP server reply to FTP client open requests. This is the default.

OFF

Specifies that the FTP server level and domain name information will not be given in the FTP server reply to FTP client open requests.

TIMESTAMP Statement

Purpose

The TIMESTAMP statement specifies whether timestamps should be shown in front of any FTP message.



Operands

ON

Specifies that timestamps should be shown in front of FTP messages.

OFF

Specifies that timestamps should not be shown in front of FTP messages.

TLSLABEL Statement

Purpose

The TLSLABEL statement specifies the TLS label used by the FTP server when securing control and data connections using TLS.



Operands

label

Specifies the TLS label to be used when securing control and data connections using TLS.

Usage Notes

Note:

1. The TLS label may be set dynamically using the FTP server MSG TLSLABEL command.
2. The TLS label can be no more than 8 characters, and must be comprised of only uppercase, alphanumeric characters.

TRACE Statement

Purpose

The TRACE statement enables FTP server tracing, which describes major actions such as beginning a dialog with a new client. Trace information can be directed to either the FILE DEBUGTRA file on the FTP server 191 minidisk or to the FTP server console.



Operands

CONSOLE

Specifies that trace information should be directed to the FTP server console. This is the default.

FILE

Specifies that trace information should be directed to the FILE DEBUGTRA file on the FTP server 191 minidisk. When TRACE FILE is used, normal FTP server activity information written to the FTP server console when the FTP server trace function is not active is still written to the console.

Usage Notes

Since the FTP server TRACE function records all FTP server activity and writes the trace information to either the FTP server console or FILE DEBUGTRA file, using the TRACE function significantly degrades FTP server performance and should only be used for debug purposes.

Step 5: Configure Automatic File Translation (Optional)

By default, the z/VM FTP server transfers files in accordance with the transfer mode and type settings specified by a client. However, when web browser FTP clients are used to interact with this server, automatic file translation (performed by the FTP server on a default basis) is recommended.

When such translation is enabled, the server performs automatic EBCDIC-ASCII translation, based on a file *extension* (or with respect to CMS files, the file *type*) of a file that is transferred using the Image transfer type. This can simplify FTP operations for various users and clients, and may be necessary to accommodate certain types of clients — web browser and graphical FTP clients, for example — since many of these clients often default to using a transfer type of Image (or, binary) and do not offer a way for users to specify a different file transfer type.

To enable default automatic file translation:

1. Specify the AUTOTRANS ON statement in the FTP server configuration file.
2. Customize the TCPIP DATA file to include the appropriate VMFILETYPE and VMFILETYPEDEFAULT statements. The FTP server relies upon these statements to control the manner in which file translation is performed for specific file extensions, as well as those that are "unknown" or not recognized. For detailed information about how to specify these statements, see [Chapter 3, "Defining the TCP/IP System Parameters,"](#) on page 13.

Note:

1. The VMFILETYPE statement determines whether EBCDIC-ASCII translation occurs for a specific file, based on the extension of that file.
2. File extensions that are not dealt with through a specific VMFILETYPE statement are considered as "unknown" (that is, are not recognized). The translation performed for such files (if any) is controlled

by the VMFILETYPEDEFAULT statement. If the VMFILETYPEDEFAULT statement is not used, unknown file extensions default to a transfer type of Image and no translation is performed.

3. The FTP server ignores the LINES operand of the VMFILETYPE and VMFILETYPEDEFAULT statements. For additional information about automatic file translation, see the [z/VM: TCP/IP User's Guide](#).

Step 6: Configure Secure FTP Connections (Optional)

The FTP server can be configured to allow, require, or restrict secure control connections and data connections using the Transport Layer Security (TLS) protocol. For more information, see RFC 4217, *Securing FTP with TLS*, which describes the protocol that is used by the FTP server.

By default, the FTP server:

- Does not allow secure control or data connections. To configure the FTP server to allow or require secure control connections, use the SECURECONTROL statement in the FTP server configuration file. To configure the FTP server to allow or require secure data connections, use the SECUREDATA statement in the FTP server configuration file.
- Does not authenticate client certificates when they connect to the FTP server. To configure the FTP server to authenticate a client certificate, use the CLIENTCERTCHECK statement in the FTP server configuration file.

Note: In order to secure FTP data connections, the control connection associated with the FTP session must be secured. Therefore, if secure data connections are desired, the FTP server must be configured to allow or require secure control connections. FTP secure connection support may also be configured dynamically using the FTP server SMSG SECURE command.

In order to support secure connections using TLS, the FTP server must have a configured TLS label. If secure connections are desired, configure the FTP server TLS label using the TLSLABEL statement in the FTP server configuration file. The TLS label may also be configured dynamically using the SMSG TLSLABEL command.

When the server is configured to support TLS and the FTP client requests a secure control connection, a negotiation will take place between the FTP client and the FTP server to secure the control connection. Once the control connection is secure, control connection transmissions are confidentiality and integrity protected by encryption. At this point, if the server is configured to support secure data connections and the FTP client requests to set up secure data connections, a negotiation will take place between the FTP client and the FTP server to secure data connections. After a data connection is secure, data connection transmissions are confidentiality and integrity protected by encryption.

The FTP server can allow, require, or restrict secure data connections on a per user basis by using the CHKIPADR exit. Please see [“Step 7: Customize FTP Server Exits \(Optional\)” on page 69](#) for more information.

Step 7: Customize FTP Server Exits (Optional)

PI

The FTP server may allow, require, or restrict secure data connections on a per user basis by using the CHKIPADR exit.

The FTP server exits are described in the following sections.

Prior to customizing the server exits described in this section, ensure that you have reviewed the exit limitations and customization recommendations presented in [“Customizing Server-specific Exits” on page 49](#).

Using the FTP Welcome Banner

The FTP server can display a site specific message when users establish a connection to the FTP server. The contents of file "FTP BANNER" will be displayed, if the file exists. When this file exists, the first such file found in the CMS search order is used. You may also choose to have one of several different banners

displayed after the user name and login password (for other than anonymous user) validation. This is handled in the CHKIPADR Exit.

Using the FTP Server Exit

The FTP server exit, FTPEXIT ASSEMBLE, can be called by the FTP server to allow greater control over how FTP commands received by this server are processed and to allow for auditing of FTP logins, logouts, and file transfers. The FTP exit is enabled using the FTAUDIT, FTCHKCMD, and FTCHKDIR configuration file statements or by using privileged SMSG commands to enable or disable the exit processes.

For audit processing (FTAUDIT enabled), the FTP exit will be called for login, logout, and data transfer events that are initiated using these FTP subcommands: APPEND, GET, PUT, DIR and LS. Information passed to the exit may be used to generate user login/logout reports and to keep track of files (and bytes) transferred in and out through the FTP server.

When the FTP server exit is enabled for general command exit processing (FTCHKCMD enabled), the FTP exit will be called to perform command validation for every received FTP command. The general command exit can be used to perform additional security checking and then take an appropriate action, such as:

- Reject commands from a particular IP address or user ID
- Reject a subset of commands for anonymous users
- Reject transfer requests for specific files
- Reject all store (APPE, STOR, STOU) commands supplied by users.

With the FTP exit enabled for CD command exit processing (FTCHKDIR enabled), the exit can validate FTP directory changes and provide greater control over access to system resources by selectively honoring or refusing a client change directory request. The exit is called when an FTP client provides one of the following commands:

- CWD or CD, to change the working directory
- CDUP, to change the working directory to the parent directory
- PASS, provided a default directory is defined in CHKIPADR EXEC for the user that supplies this command
- USER, for an anonymous login for which a default directory is defined in CHKIPADR EXEC
- APPE, DELE, LIST, NLST, RETR, SIZE, STOR, or STOU commands that involve an explicit change in directory.

Sample copies of the FTP server exit files (FTPEXIT EXEC, FTPEXIT ASSEMBLE and FTPEXIT TEXT) are supplied as softcopy files (FTPEXIT SAMPEXEC, FTPEXIT SAMPASM, and FTPEXIT TEXTSAMP, respectively) on the TCPMAINT 591 minidisk. Consult the [*z/VM: TCP/IP Programmer's Reference*](#) for details about FTPEXIT parameter list and parameter descriptions.

Using the CHKIPADR Exit

The CHKIPADR exit provides a means for controlling several aspects of FTP server processing at the time an FTP connection is initiated by the user. This capability is provided through the CHKIPADR EXEC, which is invoked by the server each time a user logs in. This exec may be used to:

- Permit or deny client access to FTP services
- Permit anonymous user login for users other than ANONYMOU
- Select a default working directory
- Select a "welcome" message, or banner
- Select a default file list format
- Select default automatic file translation

Decisions concerning these actions can be made based on the VM user ID, LOGON BY user ID, or client IP address associated with an FTP connection as it is attempted.

Providing Anonymous Login Support

Anonymous user login can be accommodated for user names other than ANONYMOU or ANONYMOUS (for which a corresponding ANONYMOU VM user ID is required). Anonymous login is permitted for user names other than ANONYMOU when return code 20 is received from CHKIPADR EXEC and when the :Anonymous . YES tag is specified in the DTCPARMS file. Anonymous users are not prompted to provide a login password.

Establishing a Default Working Directory

When a user logs in using FTP, the 191 minidisk associated with that user ID is established as a working directory, by default. However, an alternate working directory may be selected for a user when a connection is established. The alternate working directory specified may be a:

- Minidisk
- Shared File System (SFS) directory
- Byte File System (BFS) directory
- Virtual reader (RDR)
- Hardware Management Console (HMC) directory.

Providing a User-Specific Banner

A welcome message or banner can be specified for a user or group of users when a connection is accepted. Such a banner could be used to provide special instructions or supply current file/directory status information. The banner file type must be **BANNER**. The contents of the file will be displayed following user login validation. Banners specified in CHKIPADR EXEC are displayed in addition to the default **FTP BANNER** file, which is displayed at connection time.

Establishing a Default File List Format

A default file list format may be selected for a user or group of users when a connection is accepted. The selected format determines how responses to client DIR or LIST commands are initially presented. The z/VM FTP server can provide either VM-format or Unix-format file lists. The desired format default must be indicated when control is returned to the FTP server.

Note: Since many web browsers use anonymous FTP during implicit FTP transactions, a Unix-format list default is recommended for anonymous FTP clients.

Establishing Automatic File Translation Defaults

The default setting for automatic file translation, based on file extension, can be turned on or off for specific users. For detailed information, see [“Step 5: Configure Automatic File Translation \(Optional\)” on page 68](#). The desired default setting must be indicated when control is returned to the FTP server.

Note: Since many web browsers use anonymous FTP during implicit FTP transactions (and, often perform only binary file transfers), default automatic file translation is recommended for anonymous FTP clients.

Set the Minimum Security Level of Data Connections

A minimum security level for data connections may be set for a user or group of users. The selected level determines how the FTP server will respond to FTP client PBSZ and PROT commands. Secure data connections may be optional (**ALLOWED**), mandatory (**REQUIRED**), or restricted (**NEVER**). Please refer to [“Step 6: Configure Secure FTP Connections \(Optional\)” on page 69](#) for more information on configuring secure connections using TLS.

Permit Access to HMC Removable Media

Authority to ftp to HMC removable media can be granted to a user or group of users when a connection is accepted.

CHKIPADR Input

Purpose

Operands are provided to the CHKIPADR EXEC at invocation, based on the following syntax:

➔ CHKIPADR — *userid* — *ipaddress* — *byuserid* — *conn* — *ipv6_address* ➔

Operands

userid

Specifies, in uppercase, the user ID that the FTP server will use for security checking.

ipaddress

Specifies, in dotted decimal notation, the IP address that the FTP server will use for security checking.

byuserid

Specifies, in uppercase, the LOGON BY user ID if the FTP client issued a USER subcommand that included the *userid*/BY/*byuserid* operands; otherwise this field will contain a hyphen (-).

conn

Specifies the control connection number.

IPv6_address

Specifies, in colon hex notation, the IPv6 address that the FTP server will use for security checking, if any.

CHKIPADR Output

These are the return codes for the CHKIPADR EXEC:

0

User ID/IP Address is authorized

4

User ID is not authorized

8

IP Address is not authorized

12

User ID is not authorized and no error message will be sent user.

20

Anonymous user, no password required

Program stack contents upon exit may contain in any order:

- Default working directory
- Banner file name, prefixed by the keyword **BANNER**
- Translation default (**ON** or **OFF**), prefixed by the keyword **AUTOTRANS**
- List format (**VM** or **UNIX**), prefixed by the keyword **LISTFORMAT**
- Minimum data connection security level (**ALLOWED**, **REQUIRED**, or **NEVER**), prefixed by the keyword **SECUREDATA**.
- Authority to access HMC removable media (**YES** or **NO**) prefixed by the keyword **HMCAUTH**.

Example

The CHKIPADR code sample that follows causes the FTP server to perform the following actions when the user TERI initiates an FTP connection:

1. allow anonymous login (that is, TERI is not prompted for a login password)
2. establish the "server1:teri:ftp" SFS directory as the default working directory
3. initially respond to DIR commands using UNIX-format file lists
4. enable automatic file translation for **Image** file transfers
5. display the content of a WELCOME BANNER file, if it exists.

```
/* Sample processing clause for FTP user "Teri" */
When (Userid = 'TERI')
  Then Do
    Queue 'server1:teri:ftp'
    Queue 'banner welcome'
    Queue 'listformat unix'
    Queue 'autotrans on'
    status = 20
  End
```

When user TERI issues an FTP command to connect to the VMSYS1.ENDICOTT.IBM.COM host, the following responses might be produced:

```
ftp vmsys1
VM TCP/IP FTP function level 730
Connecting to VMSYS1 9.130.48.64
220-.....
.   This is the contents of FTP BANNER   .
.                                         .
.....
FTPSERV2 IBM VM function level 730 at VMSYS1.ENDICOTT.IBM.COM, 13:56:24 EST
2009-03-26
220 Connection will close if idle for more than 5 minutes.
USER (identify yourself to the host):
teri
>>>USER teri
230-.....
.   This is the contents of WELCOME BANNER   .
.                                         .
.....
230 TERI logged in; working directory = SERVER1:TERI.FTP
Command:
```

PI end

Dynamic Server Operation

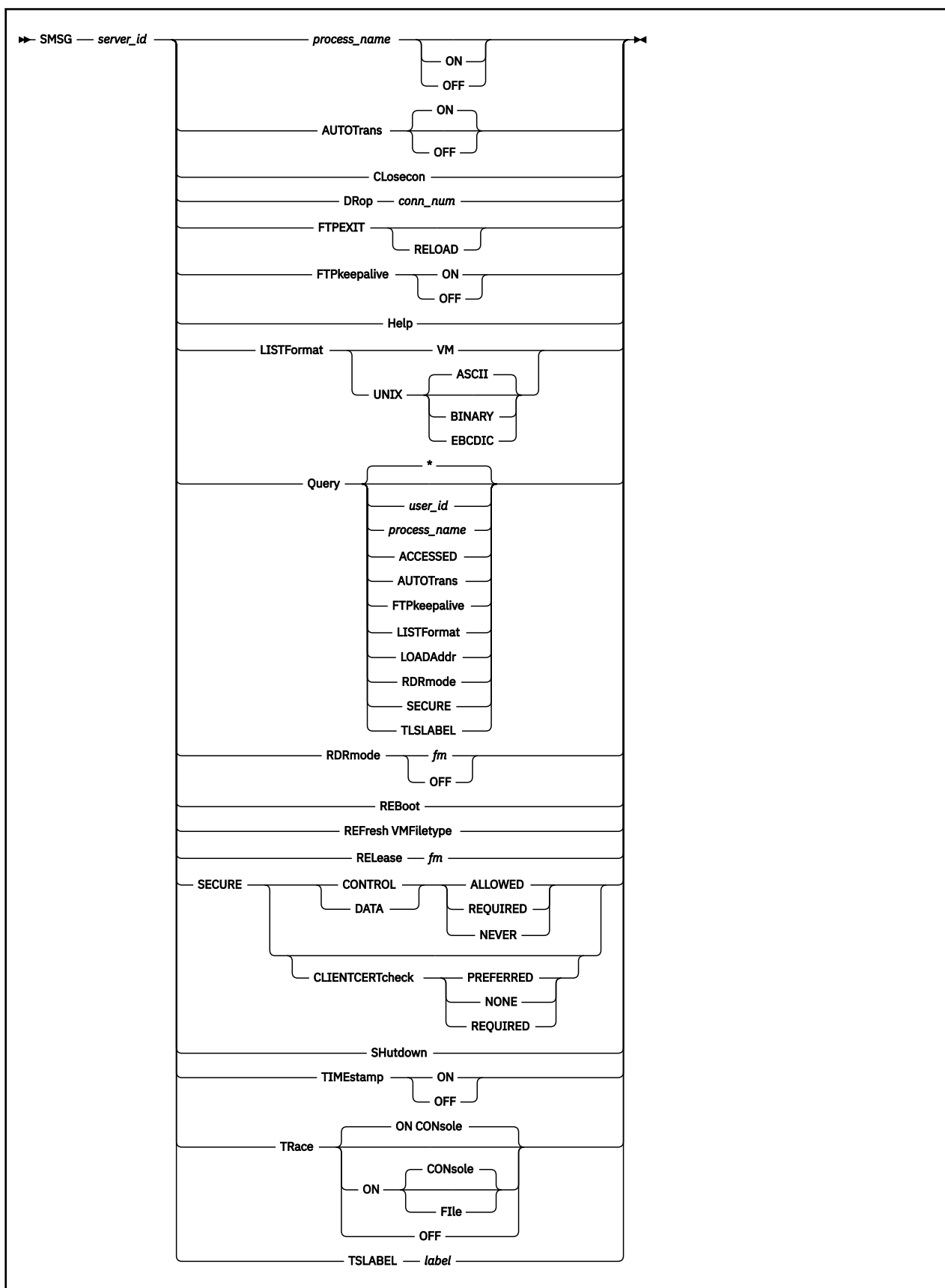
The VM Special Message Facility (SMSG) command provides an interactive interface to the FTP server to perform privileged system administration tasks, such as:

- enabling and disabling the Trace function
- querying user data
- dropping connections
- querying minidisks and directories held by the FTP server
- enabling, disabling, and querying the FTP User Exits
- setting the default list format supplied by the server
- enabling and disabling default automatic file translation.

Note:

1. Privileged SMSG commands are accepted only from users that have been included in the OBEY list of the TCPIP server configuration file.
2. Command responses are returned to the originator of the SMSG command through the use of CP MSG commands.

SMSG Interface to the FTP Server



Purpose

Use the VM Special Message Facility (MSG) command interface to the FTP server to:

- Drop connections
- Enable or disable default automatic file translation
- Enable or disable the trace function
- Enable, disable, or query the FTP user exits
- Query minidisks and directories held by the FTP server
- Query user data
- Set the behavior of requesting and authenticating the certificate of incoming FTP connections
- Set the default list format supplied by the server.

Operands

***process_name* ON | OFF**

Enables or disables the audit exit, the command exit or the CD command exit. *Process_name* may be FTAUDIT, FTCHKCMD, FTCHKDIR or FTPEXIT. If FTPEXIT is specified, all FTP exits will be enabled or disabled.

AUTOTrans ON | OFF

Determines whether automatic file translation is performed by default when files are transferred using the Image transfer type. Specify ON if automatic file translation should be enabled as the default when an FTP session is established, or OFF if the default should be to not attempt such translation. If neither ON or OFF is specified, ON is the default.

The specified default translation setting is applied to all Image transfers requested by clients unless:

- the CHKIPADR exit has been configured to select a specific translation setting when a user logs in
- the automatic translation setting is changed during an FTP session by a client via the AUTOTRANS operand of the SITE subcommand.

For more information about automatic file translation, see [“Step 5: Configure Automatic File Translation \(Optional\)” on page 68](#).

Cclosecon

Closes the FTP server console log and sends it to the :Owner. identified in the DTCPARMS file.

server_id

Specifies the FTP server user ID to which this MSG command is targeted.

Drop conn_num

Indicates the FTP server is to drop the specified connection.

FTPEXIT RELOAD

Reloads the FTP exit routine.

FTPkeepalive ON | OFF

Specifies whether or not the FTP server should make use of the TCP/IP server's keepalive mechanism to avoid timing out idle control connections. The frequency with which packets are sent is determined by the KEEPALIVEOPTIONS statement in the TCP/IP server's configuration file.

Note: Dynamic changes to the FTPKEEPALIVE setting will only affect new connections.

Help

Provides brief help about MSG commands supported by the FTP server.

LISTFormat VM | UNIX [ASCII, BINARY, EBCDIC]

Sets the format default for list information supplied by the server when it responds to client DIR (or, LIST) requests. Specify **VM** for VM-format lists to be supplied by default, or **UNIX** if the default should be Unix-format lists. With LISTFORMAT UNIX you can optionally specify the transfer mode (ASCII, BINARY, or EBCDIC) whose resulting size should be shown as file size in the output of the DIR (or,

LIST) subcommands. The default transfer mode is ASCII. The specified format default is applied to all LIST responses, unless:

- the CHKIPADR exit has been configured to select a specific format default when a user logs in
- the list format is changed during an FTP session by a client via the LISTFORMAT operand of the SITE subcommand.

For detailed information about VM-format and Unix-format responses, see the [z/VM: TCP/IP User's Guide](#).

Query *user_id*

Returns user data for the specified user or if *user_id* is omitted or is an (*), returns data for all current FTP users.

Query *process_name*

Queries the settings for the *process_name* specified. *Process_name* may be FTAUDIT, FTCHKCMD, FTCHKDIR or FTPEXIT. If FTPEXIT is specified, all FTP exit settings will be displayed.

Query **ACCESSED**

Returns CMS QUERY ACCESSED command output.

Query **AUTOTRANS**

Returns the setting (ON or OFF) that is in effect for the automatic file translation default.

Query **FTPkeepalive**

Returns the setting (ON or OFF) that is in effect for the FTP keepalive setting.

Query **LISTFormat**

Returns the setting (VM or UNIX) that is in effect for the list format default.

Query **LOADAddr**

Returns the load address of the FTP server module.

Query **RDRmode**

Returns the file mode of the resource used to temporarily store files before they are sent to a virtual reader.

Query **SECURE**

Returns the settings that are in effect for secure control and secure data connections (ALLOWED, NEVER, or REQUIRED) and client certificate checking (NONE, PREFERRED, or REQUIRED).

Query **TLSLABEL**

Returns the TLS label that is used when securing connections using TLS.

RDRmode *fm*

Sets the filemode of the resource to be used to temporarily store files before they are sent to a virtual reader. This enables the FTP server PUT support to a virtual reader. This filemode cannot be changed if the FTP server is currently processing a PUT to a reader directory. The file mode specified must be a accessed filemode to which the FTP server has Read/Write privileges.

RDRmode **OFF**

Disables the FTP server PUT support to a virtual reader. PUT to reader support cannot be disabled while the FTP server is processing a PUT to a reader directory.

REBoot

Causes the FTP server to Initial Program Load (IPL) CMS.

REFresh **VMFiletype**

Causes the FTP server to replace existing automatic file translation information with that defined by current VMFILETYPE and VMFILETYPEDEFAULT definitions in the TCPIP DATA file.

RElease *fm*

Indicates the FTP server is to issue a CMS RELEASE command for the specified file mode.

SECURE CONTROL **ALLOWED**

Secure control connections using TLS are optional. This is the default. When ALLOWED is specified, the FTP client may optionally secure the control connection using TLS; otherwise, the control connection will not be secured using TLS.

SECURE CONTROL REQUIRED

Secure control connections using TLS are required. When REQUIRED is specified, clear control connections are not allowed.

SECURE CONTROL NEVER

Secure control connections using TLS are not allowed.

SECURE DATA ALLOWED

Indicates secure data connections using TLS are optional. When ALLOWED is specified the FTP client may optionally secure data connections using TLS; otherwise, data connections will not be secured using TLS.

SECURE DATA REQUIRED

Secure data connections using TLS are required. When REQUIRED is specified, clear data connections are not allowed.

SECURE DATA NEVER

Secure data connections using TLS are not allowed.

SECURE CLIENTCERTCHECK NONE

Specifies that a client certificate will not be requested.

SECURE CLIENTCERTCHECK PREFERRED

Specifies that a client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

This is the default.

SECURE CLIENTCERTCHECK REQUIRED

Specifies that a client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

SHutdown

Initiates FTP server shutdown processing (in the same manner as the #CP EXTERNAL command) and additionally logs off the server.

TIMEstamp ON

Specifies that timestamps should be shown in front of FTP messages.

TIMEstamp OFF

Specifies that timestamps should not be shown in front of FTP messages.

TSLABEL *label*

Specifies the TLS label to be used by the FTP server when securing connections using TLS.

Note: The TLS label can be no more than 8 characters, and must be comprised of only uppercase, alphanumeric characters.

TRace OFF

Disables server tracing.

TRace ON CONsole

Enables server tracing and directs trace information to the FTP server console log.

TRace ON File

Enables server tracing and directs trace information to the FILE DEBUGTRA file on the FTP server 191 minidisk. If the trace file already exists, its previous contents are deleted.

Note: When TRACE ON FILE is used, normal FTP server activity information written to the FTP server console when the FTP server trace function is not active is still written to the console.

Usage Notes

Because the FTP server TRACE function records all FTP server activity and writes the trace information to either the FTP server console or FILE DEBUGTRA file, using the TRACE function significantly degrades FTP server performance and should be used for debugging purposes only.

Providing Web Browser FTP Support

By default, the z/VM FTP server responds to client LIST requests using VM-format lists. However, when web browser FTP clients are used to interact with this server, the use of a Unix-format list default is recommended. If VM-format file lists are supplied to such a client, that client may not correctly display or manage the supplied directory and file information, which may lead to limited or adversely affected FTP processing capabilities.

Default Unix-format lists are also recommended when graphical FTP clients are in use — again because these clients are generally Unix-based, and thus expect Unix-like information to be presented. Also, many such clients do not offer a way for users to affect file transfer operations through specific FTP subcommands; this precludes the selection of a response format.

In addition, automatic file translation may need to be enabled on a default basis, to allow for correct file translations when web browser and graphical FTP clients are in use. This is because many such browsers often default to using a transfer type of Image (or, binary) and do not offer a way for users to specify a different file transfer type. See [“Step 5: Configure Automatic File Translation \(Optional\)” on page 68](#) for more information about this topic.

Chapter 7. Configuring the LDAP Server

This topic describes how to install, configure, and run the stand-alone LDAP server and other LDAP programs.

Configuration Steps for the LDAP Server

LDAP Server Configuration Steps
1. Update the TCP/IP server configuration file (PROFILE TCPIP). 2. Update the DTCPARMS file for the LDAP server. 3. Determine the LDAP server BFS directory default. 4. Set Up the User ID and Security for the LDAP Server. 5. Copy the configuration files. 6. Create and customize the LDAP configuration file (DS CONF). 7. Set the time zone. 8. Set environment variables (DS ENVVARS). 9. Verify the LDAP server. 10. Finalize setup of LDAP backends.

Dynamic Operations: The LDAP server provides an SMSG command interface that allows you to perform various server administration tasks. For more information see [“Dynamic Server Operation”](#) on page 161.

Step 1: Update the TCP/IP Server Configuration File (PROFILE TCPIP)

Do the following:

1. Include the LDAP server virtual machine user ID in the AUTOLOG statement of the TCP/IP server configuration file. The IBM default user ID for this server is **LDAPSRV**. The user ID must be class B.
2. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
LDAPSRV  0
```

With this statement, the LDAP server is automatically started when TCP/IP is initialized.

3. Verify that the following statement has been added to your TCP/IP server configuration file:

```
PORT
389  TCP LDAPSRV          ; LDAP Server
636  TCP LDAPSRV  NOAUTOLOG ; LDAP Server (Secure)
```

The LDAP server requires these ports to be reserved for it.

4. Verify that the following statements are added to PROFILE TCPIP:

```
OBEY
LDAPSRV
ENDOBEY
```

The LDAP server must be included in the OBEY list in PROFILE TCPIP.

5. Verify that the following statement has been added to your TCP/IP server configuration file:

```
SOMAXCONN 50
```

The LDAP server requires this value for the maximum number of pending connection requests queued for its listening sockets.

Step 2: Update the DTCPARMS File for the LDAP Server

When the LDAP server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the LDAP server are:

```
:Nick.LDAPSRV      :Type.server   :Class.ldap
:Nick.ldap         :Type.class
:ESM_Enable.
:ESM_Racroute.
:Mixedcaseparms.
:Mount.
:Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

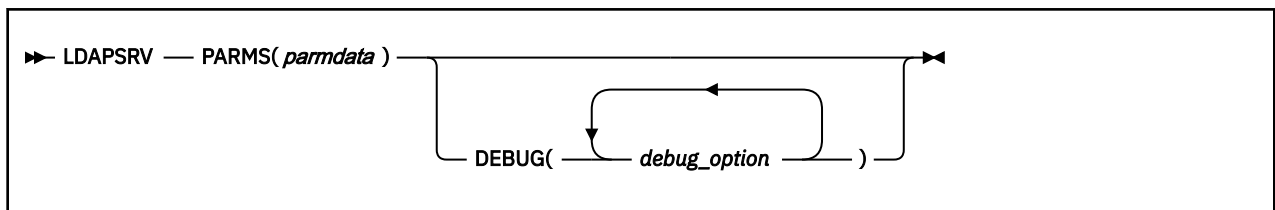
Note: You should modify the DTCPARMS file for the LDAP server if you:

- Require a root filesystem other than `/.. /VMBFS:VMSYS:ROOT/` or are using a file server other than `VMSYS`.
- Change the parameters passed to the `LDAPSRV` command. For more information about the command, see [“LDAPSRV Command Syntax” on page 80](#).
- Configure an SDBM backend or want to use LDAP server auditing. Specify `"Yes"` for `:ESM_Enable..`, and make sure `"LDAPESM"` is specified for `:ESM_Racroute..`

LDAPSRV Command Operands (:Parms. Parameters)

LDAP services are initiated using the LDAPSrv command. Operands used by this command are obtained from parameters defined by a DTCParms file :Parms. tag that is associated with an LDAP server definition. For more information about this command and its operands, see [“LDAPSrv Command Syntax”](#) on page 80.

LDAPSRV Command Syntax



Operands

PARMS(*parmdata*)

Specifies server parameters passed to the LDAP server in order to affect how the LDAP server operates. *parmdata* can include any of the parameters that are recognized by the LDAP server listed below.

Note: Unrecognized values might be ignored without notification.

-f *pathname*

Specifies the name of LDAP server configuration file to be read. The file name is case-sensitive unless it refers to an CMS file or DD statement. A CMS file is indicated by `//filename.filetype.filemode` (*filemode* is optional) and a DD statement is indicated by `//DD:dd-name`. If the `-f` parameter is not specified, the DS CONF configuration file is used.

-l *ldap_URL*

Specifies the host name or IP address and port number on which the LDAP server binds and listens for incoming requests. For information on the *ldap_URL* parameter, see *listen*. You can specify the *-l* parameter multiple times to add additional *ldap_URL* values. The values specified using the *-l* command-line parameter override the values specified for the *listen* option in the LDAP server configuration file.

You should reserve the port number or numbers chosen here in PROFILE TCPIP.

-m

Instructs the LDAP server to initialize in maintenance mode. Maintenance mode restricts the directory updates processed by the server. For more information, see ["Maintenance mode"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

DEBUG(*debug_option*)

Specifies one or more debug options that are to be activated for diagnosing problems associated with the LDAP server.

The following are the debug options. The minimum abbreviation for each option is shown in uppercase letters.

ACL	ALL	ANY
ARGS	BECApabilities	BER
CACHe	CONNs	ERROR
FILTer	INFO	LDAPBE
LDBM	MESSage	MULTIServer
OFF	PACKets	PARSe
PERFormance	REFErral	REPLication
SCHEmA	SDBM	STATs
STRBuf	TDBM (TDBM backend	THREAd
TRACe	not supported)	

Debug options other than ALL, ANY, and OFF are applied in a cumulative manner. The ALL, ANY, and OFF options, when specified, override all other debug options. For options that override other options, the last such option encountered is applied.

If the DEBUG() operand is not used, the default option in effect is OFF (no debugging options are applied).

Return codes

0

Successful execution; no errors encountered.

1

Incorrect invocation; execution cancelled.

2

Internal processing error; execution status unknown.

4

Warning(s) issued; manual review of output recommended.

8

Error(s) encountered; execution cancelled.

nn

Error(s) encountered; execution cancelled.

Step 3. Determine the LDAP Server BFS Directory Default

The LDAP server requires use of the OpenExtensions Byte File System to access the LDAP server message catalog files and to store the schema backend and other database files associated with the LDBM or

GDBM backends. The message catalog files are installed by default in `/.. /VMBFS:VMSYS:ROOT`. The working directory in which the LDAP server creates its schema and other database files defaults is `/.. /VMBFS:VMSYS:user-id/`, where *userid* is the user ID of the LDAP server.

A minimum of 1800 4K blocks must be allocated for the working directory file space.

For more information on how to install a Byte File system, see [z/VM: CMS File Pool Planning, Administration, and Operation](#).

If the message catalogs are not installed in the default filespace or you use a non-default filespace for the working directory, you must change the tag values specified for the **:Mount.** tag in the DTCPARMS file for the LDAP server.

Step 4. Set Up the User ID and Security for the LDAP Server

It is recommended that a separate user ID be established to run the LDAP server. Any user ID can be used to run the LDAP server. The examples in this topic use a user ID of LDAPSrv in the commands provided.

The user ID that runs the LDAP server must have the following attributes:

- If you have an LDBM, GDBM, or CDBM backend and the **databaseDirectory** option is specified in the LDAP server configuration file, the user ID must be able to create the specified directory if it does not already exist. If the directory exists, then the user ID must have read/write access to it.
- If the **schemaPath** option is specified in the LDAP server configuration file, the user ID must be able to create the specified directory if it does not already exist. If the directory exists, then the user ID must have read/write access to it.
- If the **schemaPath** configuration option is not specified or you have an LDBM, GDBM, or CDBM backend and have not specified the **databaseDirectory** configuration option, then the user ID must be able to create directories under **/var** or the **/var/ldap** directory must already exist. If the **/var/ldap** directory exists, then the user ID must have read/write access to the directory.
- If the **logfile** configuration option is specified in the LDAP server configuration file and it specifies a file in the BFS, the directory must already exist and the user ID must have read and write access to the directory.
- If the **logfileRolloverDirectory** configuration option is specified in the LDAP server configuration file and it specifies a file in the BFS, the directory must already exist and the user ID must have read and write access to the directory.
- If you have configured an SDBM backend, want to use native authentication, or want to use LDAP server auditing, the user ID must have authority to issue RACROUTE requests. Note that the user ID must be given UPDATE access authority to the ICHCONN facility class for proper server operation. For information on authorizing virtual machines to issue RACROUTE requests, see [z/VM: Security Server RACROUTE Macro Reference](#).
- If you have configured an SDBM backend and want to be able to log changes to a RACF user, group, or connection, add the following statement to the CP directory entry for the user ID:

```
IUCV *RPI PRIORITY MSGLIMIT 255
```

Note: To perform the RACF configuration required to support creation of LDAP change log entries, see [How to set up and use the LDAP Server for logging changes](#) in the [z/VM: TCP/IP LDAP Administration Guide](#) and [Activating LDAP Change Notification](#) in the [z/VM: RACF Security Server Security Administrator's Guide](#).

- The user ID requires assignment of a POSIX UID and GID. The IBM-provided defaults assign UID 5 and GID 0 ("system"). It is recommended that UID 0 not be used because superuser privileges are not required.
- If you are going to set up more than one LDAP server on the same system, a separate user ID should be used for each one.

Additional Setup for user ID that runs the LDAP Server

The user ID must be in the group that owns the backend directory and files or it must own the backend directory and files. If this requirement is not met, message GLD1342E is issued which indicates the UID and GIDs for the LDAP server user ID, which file or directory it does not own, and the UID and GID of the file or directory.

For example:

GLD1342E Unwilling to open file or directory '/var/ldap/schema':

File or directory UID 8, UID of program 0, GID of file or directory 8, GIDs of program (10, 0, 1, 110011).

In the message text:

- The file or directory is "/var/ldap/schema". An "ls -n" of this path shows that it is a directory.
- The owning UID of the directory is 8.
- The owning GID of the directory is 8.
- The UID of the program is 0. Therefore, it does not own the directory.
- The GIDs of the program are 10, 0, 1 and 110011. Therefore, it is not in a group that owns the directory.

If the server has other file-based backends such as CDBM, LDBM, or GDBM, then an "ls -n" of the backend directory (as specified in the **databaseDirectory** option in the LDAP server configuration file) shows the UID and GID of the files and directories. All the backend files and directories must be owned by the user ID that runs the LDAP server, or be owned by one of the user ID's groups.

The z/VM OpenExtensions chown command can be used to change the owner of the file or directory. The z/VM OpenExtensions chgrp command can be used to change the owning group of the file or directory.

Note: When the LDAP server creates the files, the owning group is inherited from the parent directory.

These requirements also apply to the ds2ldif utility. The ds2ldif utility accesses the schema directory and file, and the directory and files for the backend being unloaded. Therefore, the user ID that runs the utility must be in the group that owns these directories and the files within these directories.

Additional Setup for Auditing

If you plan to generate LDAP SMF 83 audit records or want to use remote auditing, the following RACF commands must be entered to set up the user ID that will run the LDAP server:

```
RDEFINE FACILITY IRR.RAUDITX UACC(NONE)
PERMIT IRR.RAUDITX CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
SETOPTS RACLIST(FACILITY) REFRESH
```

Additional setup is required for remote auditing, see [Setting up authorization for working with remote services in Remote authorization and auditing through LDAP](#) of the *z/VM: TCP/IP Programmer's Reference*.

Step 5. Copy the Configuration Files

A sample LDAP configuration file (LDAPDS SCONFIG) and a sample LDAP environment variables file (LDAPDS SAMPENV) are provided on the TCPMAINT 591 disk.

1. If you have not already done so, copy your customized LDAP configuration file to the TCPMAINT 198 minidisk as DS CONF.
2. If you have not already done so, copy your customized LDAP environment variables file to the TCPMAINT 198 minidisk as DS ENVVARS.

Step 6. Create and Customize the LDAP Configuration File (DS CONF)

The LDAP server requires one configuration file, DS CONF. This file is used to specify operational parameters for the LDAP server virtual machine.

This topic contains information on how to set up the LDAP server configuration file and how to configure the LDAP server to run with the options you choose. To keep the initial configuration simple, this step

explains how to configure the LDBM backend and native authentication. At the end of this step, you will find links to information about other backends, SSL/TLS, remote services, and encryption.

- [“Creating the DS CONF File” on page 84](#)
- [“LDAP Server Operational Mode ” on page 84](#)
- [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#)
- [“Setting up for LDBM” on page 87](#)
- [“Native Authentication” on page 88](#)
- [“Configuring Other Backends, SSL/TLS, Remote Services, and Encryption” on page 99](#)

Creating the DS CONF File

A sample LDAP server configuration file is provided as LDAPDS SCONFIG on the TCPMAINT 591 disk. The initial configuration contains default versions of some configuration settings. It does not contain a database suffix.

1. Copy your customized LDAP server configuration file to the TCPMAINT 198 minidisk as DS CONF.
2. If you want to specify a configuration file other than the default DS CONF, use the **-f** command-line parameter when starting the LDAP server.

For specifics on the configuration file options, see [“Configuration File \(DS CONF\) Format and Configuration Options” on page 127](#).

LDAP Server Operational Mode

The LDAP server may be configured to run in one of several operational modes when an LDBM or GDBM backend is configured.

• Single-server mode

In this operational mode, only a single instance of the LDAP server may use a given LDBM or GDBM database to store directory data. This server may perform replication of LDBM database changes to other servers (on the same host system or on another host system).

• Multiple single-server mode LDAP servers

In this operational mode, two or more LDAP servers, each in single-server mode, can be run on the same system with different LDBM or GDBM backends. These servers may perform replication of LDBM database changes to other servers (on the same host system or on another host system). However, each server must have its own separate replica.

Restriction: The LDAP servers cannot share LDBM or GDBM databases and cannot share the LDAP server schema. This means that each server must have unique values for the **databaseDirectory** configuration option (for an LDBM or file-based GDBM backend), and the **schemaPath** configuration option (for the schema).

In either of these modes, all combinations of LDBM (one or more), SDBM, and GDBM backends are supported.

Note: A single LDAP server instance can have one SDBM backend and one GDBM backend, but it can have multiple LDBM backend instances.

For more information about replication, see [Basic replication in *z/VM: TCP/IP LDAP Administration Guide*](#).

For an example of specifying the DTCPARMS and PROFILE TCP/IP files for multiple single-server mode LDAP servers, see [“Duplicating and Running Existing Servers” on page 44](#).

Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords

There are several ways that the LDAP root administrator and replica server distinguished names and password values can be configured. One of these ways must be used because an LDAP root administrator DN and password are required for the LDAP server and some other LDAP directory programs to operate:

- A root administrator DN must be specified in the global section of the configuration file using the **adminDN** configuration option (see [adminDN](#)). The password for this root administrator DN can optionally (this is not recommended) be placed in the configuration file using the **adminPW** configuration option (see [adminPW](#)) or can be held in the namespace managed by this instance of the LDAP server.
- If a replica is being established for a backend, the **masterServerDN** or **peerServerDN** configuration option must be specified in the backend section of the configuration file. The **masterServerPW** or **peerServerPW** configuration option can optionally be specified. (This is also not recommended.) All the options described below are applicable for **adminDN** and the first three options described below are applicable for **masterServerDN** and **peerServerDN**.

Note: The LDAP root administrator can delegate server administrator responsibility by defining the administrative group, adding members, and assigning administrative roles. Administrative roles can be defined in LDAP by an LDAP root administrator or in RACF by a RACF administrator. See [Administrative group and roles in z/VM: TCP/IP LDAP Administration Guide](#) for more information.

- Root administrator DN and password in configuration file

The simplest but least secure method is to select a root administrator DN that is outside of the scope of suffixes managed by this server (see [suffix](#)) configuration option. In other words, choose a root administrator DN such that it does not fall within the portion or portions of the namespace managed by this server. Selection of this type of root administrator DN requires that the password be placed in the configuration file using the **adminPW** configuration option (see [adminPW](#)). There is no password policy support for the root administrator DN when the password is defined in the configuration file.

For example, you might choose a simple DN, such as "cn=Admin" for the root administrator DN and a simple password such as `secret`. The configuration file options would then be established this way:

```
adminDN "cn=Admin"
adminPW secret
```

Note: Do not use the example above without changing the password value, as well as the actual distinguished name.

When a program or user binds using this root administrator DN, the LDAP server verifies that the password supplied on the request matches the value provided in the configuration file for the **adminPW** option.

Note: When first configuring an LDBM, or CDBM backend, it might be necessary to use this approach until the schema supporting the directory entries is loaded. When the schema is loaded and the entry representing the root administrator is added, the **adminDN** can be changed to the entry DN (see the next list item regarding "Root administrator DN and password as an LDBM, or CDBM entry"). The server must be restarted to pick up the new **adminDN**.

- Root administrator DN and password as an LDBM or CDBM entry

In this method, the root administrator DN is established as an entry managed by an LDBM or CDBM backend. The **userPassword** attribute is used to hold the password for the root administrator DN in this case. There is LDAP password policy support for the root administrator DN when the entry contains a **userPassword** attribute value in the LDBM or CDBM backend.

Alternatively, the password or password phrase can be stored in RACF if native authentication is configured. The LDAP password policy support does not apply to the root administrator DN when the password or password phrase resides in RACF. RACF provides the authentication rules and enforcement of its own password policy. For more information on using native authentication, see ["Native Authentication" on page 88](#).

For example, if the LDBM backend is managing the portion of the namespace "o=Your Company", one root administrator DN that could be selected is "cn=LDAP Admin,o=Your Company".

The configuration file would include the following options:

```
adminDN "cn=LDAP Admin,o=Your Company"
...
database ldbm GLDBLD31
...
suffix "o=Your Company"
```

The LDIF-format entry to be added to the database through LDAPADD might be:

```
dn: cn=LDAP Admin,o=Your Company
objectclass: person
cn: LDAP Admin
description: Administrator DN for o=Your Company server
sn: Administrator
uid: admin
userpassword: secret
```

Note: Do not use the example above without changing the password value, as well as the actual distinguished name.

If this entry is contained in a file system file called `admin.ldif`, it can be loaded using LDAPADD:

```
ldapadd -h ldaphost -p ldapport -D binddn -w passwd -f admin.ldif
```

Note: The `ldapadd` example above assumes that the LDAP server is running and that the suffix entry (entry with the name "o=Your Company") exists. Furthermore, the `binddn` is assumed to exist and have sufficient authority to add the entry. When initially setting up the LDAP server, one way to satisfy the assumption is to first configure the LDAP server using the `adminDN` and `adminPW` configuration options. Then, start the LDAP server, load the suffix entry and the root administrator DN entry, using the `adminDN` and `adminPW` configuration values for `binddn` and `passwd` respectively. After the add operations complete, stop the LDAP server, change the `adminDN` configuration option value to the name of the entry just added and **remove** the `adminPW` configuration option. Then restart the LDAP server.

When a program or user binds using this root administrator DN, the LDAP server verifies that the password supplied on the request matches the value of the **userPassword** attribute stored in the entry.

CRAM-MD5 and DIGEST-MD5 authentication binds with the **adminDN** are supported as long as the entry exists in an LDBM or CDBM backend. The **adminDN** entry must contain a **uid** attribute value that will be used as the user name by a client application when attempting a CRAM-MD5 or DIGEST-MD5 authentication bind. For more information on CRAM-MD5 and DIGEST-MD5 authentication, see [CRAM-MD5 and DIGEST-MD5 Authentication in *z/VM: TCP/IP LDAP Administration Guide*](#).

- Root administrator DN and password in RACF

This method requires that the LDAP server be configured to use the RACF support provided in the SDBM backend. The root administrator DN can be established as a RACF-style DN based upon a RACF user ID. (For more information, see [RACF-style distinguished names in *z/VM: TCP/IP LDAP Administration Guide*](#).)

In this case, the password for the root administrator DN is the RACF user ID's password or password phrase, and is stored and verified by RACF. The LDAP password policy support does not apply to the root administrator DN when the password or password phrase is stored in RACF. RACF provides the authentication rules and enforcement of its own password policy.

For example, if you configure the LDAP server with RACF support where the portion of the namespace held by RACF is "o=Your Company", and the RACF user ID that is used for the administrator is `g1admin`, the configuration file would include these options:

```
adminDN "racfid=g1admin,profiletype=user,o=Your Company"
...
database sdbm GLDBSD31
suffix "o=Your Company"
```

When a program or user binds using this root administrator DN, the LDAP server makes a request to RACF to verify that the password supplied on the request matches the RACF password or password phrase for RACF user ID `g1admin`.

Setting up for LDBM

The LDAP server provides a file-based backend to store directory information in a Byte File System. LDBM is a general-purpose backend that can store any type of directory information.

The amount of space needed to store an LDBM backend in an Byte File System is approximately four to six times the size of the expected input LDIF data. Generally, the space required to hold the LDBM backend data is two to three times the size of the expected input LDIF data. However, during the LDBM commit process each of the LDBM database files is copied, therefore, resulting in occasionally needing twice the amount of Byte File System space.

When the LDAP server starts for the first time with a new LDBM backend configured, the server automatically creates the directories specified in the **databaseDirectory** server configuration option (or in `/var/ldap/ldbm` if the configuration option is not specified). When the directories are created, the LDAP server's userid is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP server's userid. The group that the LDAP server's userid belongs to is granted read and write access to the directories. As part of the LDBM backend initialization process, the server creates **LDBM-x.db** files (where *x* is a number such as 1, 2, 3, and so on) for each suffix in the LDBM backend section, along with an **LDBM.ckpt** file. These files are created with the LDAP server's userid as the owner. The default permissions on these files grant read and write access to both the LDAP server's userid and the group to which the LDAP server's userid belongs.

If the default LDBM backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information on the **chmod** and **chown** commands, see [z/VM: OpenExtensions Commands Reference](#).

In order to configure your LDAP server to run with the LDBM backend of the LDAP server:

1. If you have not already done this, copy the configuration files from the TCPMAINT 591 disk (see [“Step 5. Copy the Configuration Files”](#) on page 83).
2. You need to use the following lines in your DS CONF file:

```
database ldbm GLDBLD31
suffix "your_suffix"
```

where *your_suffix* is any valid DN (distinguished name). Be sure to provide a meaningful value for the suffix.

Notes:

- a. Multiple LDBM backends can be configured in an LDAP server, but each must use a different file directory for storing its entries. If you are configuring multiple LDBM backends or do not want an LDBM backend to store its entries in the default file directory, then add the **databaseDirectory** option to the LDBM section of the configuration file. The **databaseDirectory** value must be different than the ones used by any other LDBM, GDBM, or CDBM backend. The LDAP server must have read-write access to the file directory. See [“Step 4. Set Up the User ID and Security for the LDAP Server”](#) on page 82.
- b. The attributes and object classes used by LDBM depend on your usage of LDBM. You will probably have to add schema to the LDAP server schema. For more information on adding schema to the LDAP server, see [Setting up the schema for LDBM and CDBM](#) in [z/VM: TCP/IP LDAP Administration Guide](#).
- c. The files contained in the directory specified by the **databaseDirectory** server configuration option are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences could occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

For information about setting up the LDAP server schema, see [LDAP directory schema in z/VM: TCP/IP LDAP Administration Guide](#).

Copying an LDBM Database

If you want to copy an existing LDBM database to a new one, you should use DS2LDIF to unload the existing LDBM database. Then use the LDAPADD command to load the new LDBM database. If you want to retain the existing **ibm-entryuuid** attribute values for the entries, the LDAP server must be put into maintenance mode. For more information, see [Basic replication maintenance mode in z/VM: TCP/IP LDAP Administration Guide](#).

If the new LDBM database is not on the same LDAP server as the existing one, the LDAP server schema for the target LDAP server needs to contain all the attributes and object classes used by the LDBM entries before they can be loaded into the new LDBM database. You can use the DS2LDIF or LDAPSrch commands to unload the LDAP server schema from the source LDAP server and then load the schema LDIF file into the target LDAP server using the LDAPMDFY command.

Native Authentication

The z/VM LDAP server has the ability to authenticate to the Security Server through the LDBM or CDBM backend by specifying a Security Server password or password phrase on a simple bind to the backend. Authorization information is still gathered by the LDAP server based on the DN that performed the bind operation. The LDAP entry that contains the bind DN should contain either the **ibm-nativeId** or **uid** attribute to specify the Security Server ID that is associated with this entry. The ID and password or password phrase are passed to the Security Server and the verification of the password or password phrase is performed by the Security Server. Another feature of native authentication is the ability to change your password or password phrase on the Security Server by issuing an LDAP modify command.

Note:

1. The SDBM backend does not have to be configured in order to use native authentication.
2. After a successful native authentication bind, the bound user can send LDAP requests to any of the configured backends. If SDBM is configured, SDBM operations are performed under the context of the Security Server ID that was used during the native authentication bind. For all other backends, LDAP operations are performed using the normal bind information (the bind DN and the groups to which it belongs).

Initializing Native Authentication

To enable native authentication, perform the following steps:

1. Install and configure RACF or another Security Server.
2. Configure an LDAP server to run with an LDBM or CDBM backend and then start the server. Specify the native authentication options in your LDAP server configuration file. For example:

```
useNativeAuth SELECTED
nativeAuthSubtree o=IBM,c=US
nativeAuthSubtree o=Lotus,c=US
nativeUpdateAllowed ON
```

3. Be sure that the entries that are to perform native authentication contain either the **ibm-nativeId** attribute or a single-valued **uid** attribute with the appropriate Security Server ID as its value. It is important to note that a multi-valued **uid** without an **ibm-nativeId** causes the bind to fail because the LDAP server does not know which ID to use.

Schema for Native Authentication

The LDAP server schema always contains the schema elements needed for native authentication. No additional schema is needed.

Following is the native authentication attribute type:

ibm-nativeId

Specifies the Security Server ID that is to be associated with this entry.

Following is the native authentication object class:

ibm-nativeAuthentication

Allows specifying the **ibm-nativeId** attribute in entries.

Defining Participation in Native Authentication

There are many different configuration options for native authentication that are discussed in this section.

The main configuration option, **useNativeAuth**, can be set to **selected**, **all**, or **off**. If you want all entries in a certain subtree to participate in native authentication then you would choose **all** for this option. However, if you would like specific entries in the specific subtrees to be subject to native authentication, then choose **selected** for the **useNativeAuth** option. When **selected** is used, only entries with the **ibm-nativeId** attribute are subject to native authentication.

Next, consider what portions of your directory should have the ability to participate in native authentication. If the entire directory should participate, then set the **nativeAuthSubtree** configuration option to **all**. If there are different subtrees in your directory which contain entries that need to bind natively or perform native password or password phrase modifications, then you need to list all the subtrees with multiple **nativeAuthSubtree** configuration options.

Note: If the DN that is listed in the **nativeAuthSubtree** options contains a space character in it, then the entire DN must be enclosed in quotation marks in the LDAP server configuration file.

In order for an entry to bind natively or perform a native password or password phrase modify, that entry must contain a mapping to the Security Server identity that is associated with the user. This can be accomplished by using either the **ibm-nativeId** attribute or the **uid** attribute. If your directory entries already contain a single-valued **uid** attribute (which holds the Security Server user ID), then these entries are already configured for native authentication if you plan on using the **useNativeAuth all** option. If you do not plan on using **uids** for mapping, then you can specify the **ibm-nativeId** attribute for your Security Server ID associations and this attribute is used with **selected** or **all** specified for the **useNativeAuth** option. If both the **ibm-nativeId** and **uid** attributes exist in an entry, the **ibm-nativeId** value is used. If a native entry has an existing **userPassword** attribute value because it was originally created under a non-native authentication subtree and the Security Server identity that is specified has not yet been defined in the Security Server, the LDAP server attempts an LDAP simple bind.

If you use the **useNativeAuth** option, also specify the **nativeUpdateAllowed** option to enable native password or password phrase changes in the Security Server to occur through the LDBM or CDBM backend.

An entry that is participating in native authentication cannot normally contain the **userPassword** attribute. An LDAP add request of an entry that contains a **userPassword** attribute value fails. An LDAP modify request that enables an entry for native authentication removes any existing **userPassword** attribute values for the entry.

Binding with Native Authentication

As mentioned above, there are two LDAP operations affected: bind and password or password phrase modify. There is a set of criteria that is used to determine if an entry actually participates in native authentication. This criteria changes depending on the configuration options that have been selected. The following table outlines all the possible operating modes for native authentication binding.

Table 11. Operating modes for native authentication binding

Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Bind	selected	any value	User1		Entry is configured correctly and native authentication is attempted.

Table 11. Operating modes for native authentication binding (continued)

Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Bind	selected	any value		User1	Entry is not correctly configured for native authentication so an LDAP simple bind is attempted. The uid attribute is not used when useNativeAuth is selected .
Bind	selected	any value			Entry has not been configured for native authentication so an LDAP simple bind is attempted.
Bind	all	any value	User1	User2	The ibm-nativeId attribute is used to attempt native authentication.
Bind	all	any value		User1	Entry is configured correctly and native authentication is attempted.
Bind	all	any value			For ease of implementation, a LDAP simple bind is attempted, even though you have specified that all entries should use native authentication. This entry should be configured correctly.

Notes: This table assumes that the entry is located within native authentication subtrees.

In native authentication binding, the LDAP server invokes **__passwd()** using the mapped user ID and the password or password phrase supplied in the bind request. The following *errno* values returned by **__passwd()** have an LDAP reason code defined for them:

Table 12. The *errno* values returned by **__passwd()** when binding

<i>errno</i> value	Reason	Text
EACCES	R004111	The password is not correct
EMVSERR	R004107	The __passwd function failed; not loaded from a program controlled library
EMVSSAF2ERR (system problem)	R004176	The __passwd() function failed with error <i>errno</i> . Reasons for this <i>errno</i> are that the Security Server is not active or there is a problem with LDAPSrv authorization to the ICHCONN profile in the FACILITY class.
EMVSEXPIRE	R004109	Password has expired
EMVSSAFEXTRERR	R004110	User ID has been revoked
EINVAL	R004112	A bind argument is not valid
EMVSSAF2ERR (userid problem)	R004108	Native user ID 'userid' is not defined
EMVSPASSWORD	R004108	Native user ID 'userid' is either not defined or no UID is present in the OMVS segment
ESRCH	—	Attempts a simple bind

Note: The same reason codes are issued when binding with a password or a password phrase.

More detailed information on **__passwd()** error codes is available from *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

Updating native passwords and password phrases

A native password or password phrase modify can be performed in one of two ways:

1. Using the LDAPMDFY utility to delete the current **userPassword** attribute followed by an add of the new **userPassword** value.
2. Using the LDAPCHPW utility to automatically delete the current **userPassword** value followed by an add of the new **userPassword** value.

Note: The **userPassword** attribute is used as a mechanism to change the native password or password phrase, but an entry that is using native authentication cannot actually include the **userPassword** attribute. An add request of an entry fails if it contains the **userPassword** attribute. A modify request of an entry will remove any existing **userPassword** attribute values from the entry. You cannot issue a single delete, add, or replace of **userPassword** values; you can specify only the combination of delete followed by an add.

If using the LDAPMDFY utility, specify the current password or password phrase on the delete statement followed by the new password or password phrase on the add statement. The delete must occur before the add for native password or password phrase modify. For example, if the file `pw.mod` contains:

```
cn=You,o=IBM,c=US
-userpassword=currentpassword
+userpassword=newpassword
```

then the following command modifies the native password or password phrase (assuming the bind DN has the authority to do this) with the LDAPMDFY utility:

```
ldapmdfy ... -D cn=You,o=IBM,c=US -w currentpassword -f pw.mod
```

If using the LDAPCHPW utility, specify the current password or password phrase on the **-w** option and the new password or password phrase on the **-n** option. For example, the following command modifies the native password or password phrase (assuming the bind DN has the authority to do this) with the LDAPCHPW utility:

```
ldapchpw .... -D cn=You,o=IBM,c=US -w currentpassword -n newpassword
```

For more information about the LDAPCHPW utility, see [z/VM: TCP/IP User's Guide](#).

An error is returned if the user ID specified by the **ibm-nativeId** or **uid** attribute value is not defined to RACF. Also, the current and new **userPassword** values must both be passwords or password phrases. An error is returned if one of the values is a password and the other is a password phrase.

The following table outlines all the possible operating modes for native authentication password updates. The same operating modes and behaviors also apply to native authentication password phrase updates.

Table 13. Operating modes for updating native password or password phrases				
Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId uid	Behavior
Modify-Replace (password)	selected	Yes	User1	Operation is not allowed because the entry is configured for native authentication. A modify-delete followed by a modify-add must be performed.
Modify-Replace (password)	selected	Yes		Entry is not configured for native authentication so a regular LDAP password replace is attempted.

Table 13. Operating modes for updating native password or password phrases (continued)					
Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Modify-Replace (password)	all	Yes			Operation is not allowed. modify-delete followed by a modify-add must be performed.
Modify-Delete (password)	selected	Yes	User1		Entry is configured for native authentication so the value specified is used to change User1's Security Server password if a modify-add follows this operation. If a modify-add does not follow, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Delete (password)	selected	Yes			Entry is not configured for native authentication so a regular LDAP modify-delete is attempted.
Modify-Delete (password)	all	Yes	User1	User2	Entry is configured for native authentication so the value specified is used to change User1's Security Server password if a modify-add follows this operation. If a modify-add does not follow, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Delete (password)	all	Yes		User1	Entry is configured for native authentication so the value specified is used to change User1's Security Server password if a modify-add follows this operation. If a modify-add does not follow, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Delete (password)	all	Yes			A regular LDAP modify-delete is allowed in this case to allow for old LDAP passwords stored in LDBM or CDBM to be removed.
Modify-Add (password)	selected	Yes	User1		If a password modify-delete was previously performed, then a Security Server password change for User1 is attempted. If the modify-delete has not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Add (password)	selected	Yes			Entry is not configured for native authentication so a regular LDAP modify-add is attempted.

Table 13. Operating modes for updating native password or password phrases (continued)					
Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Modify-Add (password)	all	Yes	User1	User2	If a password modify-delete was previously performed then a Security Server password change for User1 is attempted. If the modify-delete has not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Add (password)	all	Yes		User1	If a password modify-delete was previously performed, then a Security Server password change for User1 is attempted. If the modify-delete has not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify-Add (password)	all	Yes			Operation fails because the entry is not correctly configured for native authentication.
Add (entry with password)	selected		User1		Entry is configured for native authentication so adding an entry with a password is not allowed.
Add (entry with password)	selected			User1	Entry is not configured for native authentication so the operation is attempted.
Add (entry with password)	selected				Entry is not configured for native authentication so the add operation is attempted.
Add (entry with password)	all		User1	User2	Operation fails. Native entries cannot contain LDAP passwords.
Add (entry with password)	all			User1	Operation fails. Native entries cannot contain LDAP passwords.
Add (entry with password)	all				Operation fails. Native entries cannot contain LDAP passwords.
Note: This table assumes that the entry is located within native authentication subtrees.					

To update a native password or password phrase, the LDAP server invokes **__passwd()** using the mapped user ID and the old and new passwords or password phrases supplied in the modify delete/add request. The following *errno* values returned by **__passwd()** have an LDAP reason code defined for them:

Table 14. The <i>errno</i> values returned by __passwd() when updating password or password phrase		
<i>errno</i> value	Reason	Text
EACCES	R004111	The password is not correct

Table 14. The *errno* values returned by `__passwd()` when updating password or password phrase (continued)

errno value	Reason	Text
EINVAL	R004112	A bind argument is not valid
EMVSERR	R004107	The <code>__passwd</code> function failed; not loaded from a program controlled library
EMVSEXPIRE	R004109	The password has expired
EMVSPASSWORD	R004128	Native authentication password change failed: The new password is not valid, or does not meet requirements
EMVSSAFEXTRERR	R004110	The user id has been revoked
EMVSSAF2ERR (system problem)	R004176	The <code>__passwd()</code> function failed with error <i>error_code</i> . Reasons for this <i>error_code</i> are that RACF is not active or there is a problem with LDAPSRV authorization to the ICHCONN profile in the FACILITY class.
EMVSSAF2ERR (userid problem)	R004108	Native user ID <i>name</i> is not defined.
ESRCH	R004118	Native user ID <i>name</i> is not defined.

Note: The same reason codes are issued when updating a password or a password phrase.

The return code returned by LDAP is **LDAP_OPERATIONS_ERROR** when the *errno* value is EMVSERR, EMVSSAF2ERR (system or userid problem), or ESRCH. For the other *errno* values, the return code is **LDAP_INVALID_CREDENTIALS**.

More detailed information on `__passwd()` error codes is available from *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

Updating native passwords or password phrases during bind

Note: This section applies only to changing native passwords during bind. This method cannot be used to change the **userPassword** value during a bind to an LDBM or CDBM entry that does not use native authentication.

It is also possible to change the RACF password or password phrase of an LDBM or CDBM entry participating in native authentication during an LDAP simple bind. This may be necessary if the LDAPMDFY utility above fails with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code:

R004109 The password has expired

The simple bind occurs as part of an LDAP function such as search, compare, add, or modify. The password or password phrase change is provided in the password portion of the LDAP simple bind. The change must be in the following format:

currentvalue/newvalue

The current value and the new value must both be passwords or both be password phrases. An error is returned if one of the values is a password and the other is a password phrase.

The forward slash (/) is used as the indication of a password or password phrase change during the LDAP simple bind. Password or password phrase changes made using the LDAP simple bind to an LDBM or CDBM entry participating in native authentication are subject to the system password or password phrase rules. If the new password or password phrase does not pass the rules established on the system, a

password or password phrase change fails with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code of:

```
R004128 Native authentication password change failed: The new password is not valid,  
or does not meet requirements
```

Note: A forward slash (/) is a legal character in a password phrase (but not in a password). During native authentication bind, a backward slash (\) is an escape character to indicate the next character is part of the password or password phrase and has no special meaning. The backward slash is removed during bind processing. Therefore, during bind, a forward slash in a password phrase must be preceded by a backward slash (\) to indicate that the forward slash is part of the password phrase and is not the password phrase change indicator. For example, the password phrase `this1slash/` is part of the value `2use` must be specified as `this1slash\` is part of the value `2use` during bind. A backward slash is a legal character in a password phrase (but not in a password). Therefore, a backward slash in a password phrase must be preceded by another backward slash to indicate that it is not an escape character.

Once the bind succeeds, the password or password phrase is changed even if the LDAP function eventually fails. The **nativeUpdateAllowed** server configuration option setting does not control whether or not password or password phrase modifications can occur on an LDAP bind operation. The setting of **nativeUpdateAllowed** only controls password or password phrase modifications on a LDAP modify operation.

Assuming an LDBM or CDBM entry `cn=USER1,ou=END,o=IBM,c=US` is participating in native authentication and is mapped to user ID `USER1`, the following command changes the RACF password for user `USER1` from `abc` to `def`:

```
ldapsrch -h ldaphost -p ldapport -D "cn=USER1,ou=END,o=IBM,c=US" -w abc/def -b \  
"ou=END,o=IBM,c=US" "objectclass=*
```

Password policy with native authentication

When authenticating with a user in an LDBM or CDBM backend that is participating in native authentication or doing the special delete-add modification of the bound user's **userPassword** attribute value, the password policy applied is determined by RACF. Therefore, any configured LDAP password policy does not apply in these scenarios.

When the **PasswordPolicy** control is sent on a bind or modify request, the **PasswordPolicy** response control is returned on the bind and modify response and has additional warning and error information about the authenticating user's password or the updating of the native password or password phrase with the special delete-add modification operation. Based on information returned from RACF, only the following **PasswordPolicy** response control error codes are supported: **accountLocked**, **changeAfterReset**, **insufficientPasswordQuality**, **mustSupplyOldPassword**, **passwordExpired**, and **passwordModNotAllowed**. For more information, see [Password policy in z/VM: TCP/IP LDAP Administration Guide](#).

Also, note that a native authentication bind using an expired native password succeeds, as long as the **PasswordPolicy** control is included in the bind request and the **nativeUpdateAllowed reset** configuration file option is specified. After the bind, only the special delete-add modification of the bound user's **userPassword** attribute can be performed to reset the native authentication password. After completion, other LDAP operations can be performed.

Example of Setting up Native Authentication

The following diagram shows an example of how you could set up native authentication.

Note: Due to space limitations of the diagram, the entries in the example do not contain all of the necessary information to make them valid directory entries. For example, object classes and required attributes have been left out of many of the entries.

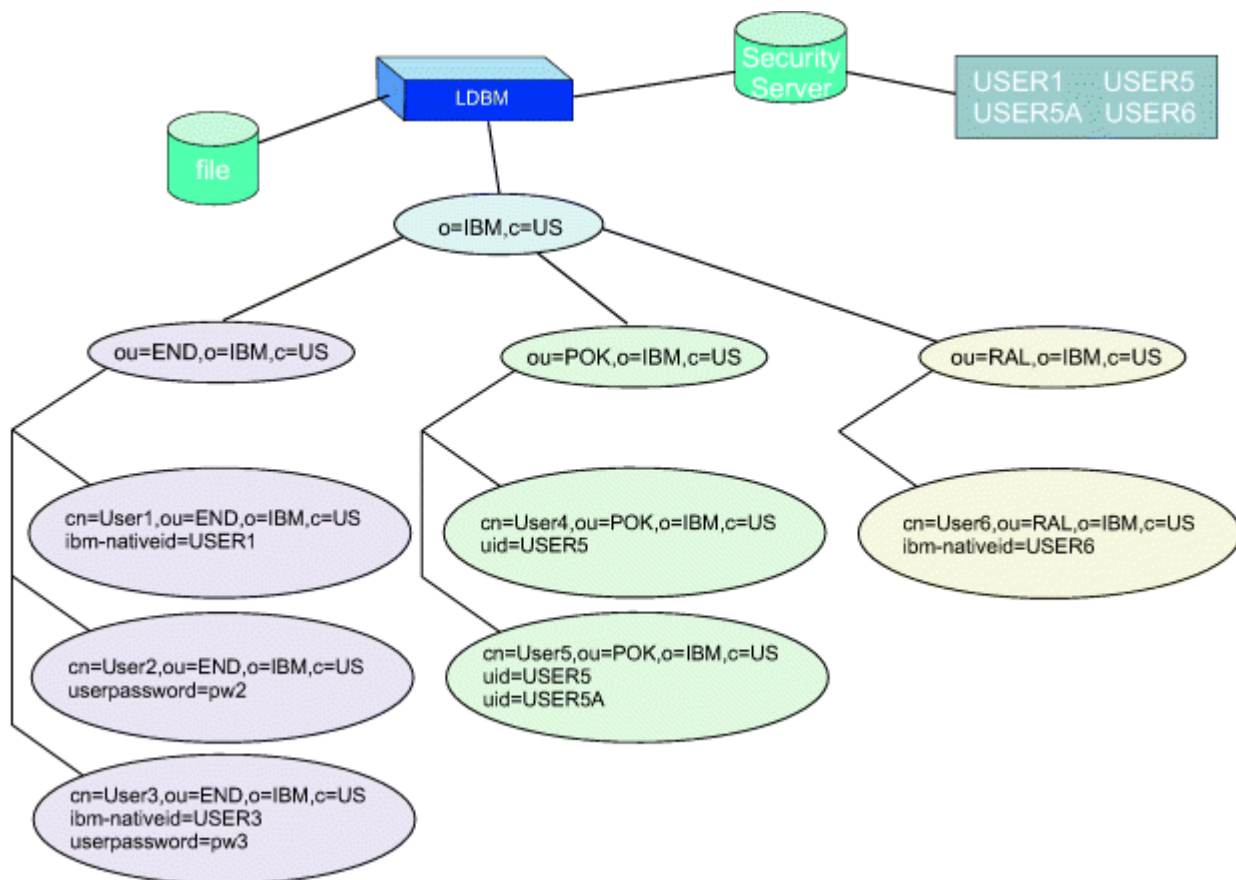


Figure 1. Native authentication example

Note: In the behavior table for each of the following examples, a password phrase can be used instead of a password, with the same results.

Example 1

- Assuming these settings:

useNativeAuth selected

nativeUpdateAllowed on

nativeAuthSubtree ou=END,o=IBM,c=US

nativeAuthSubtree ou=POK,o=IBM,c=US

the following table indicates the results of operations involving each user entry:

Table 15. Behavior of native authentication in example 1

LDAP entry	Operation	Behavior
cn=User1,ou=END,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid ibm-nativeId .
	Bind with native password change	Can change this native password because the entry contains a valid ibm-nativeId .
	modify-delete and modify-add (userPassword)	Can change this native password because the entry contains a valid ibm-nativeId .

Table 15. Behavior of native authentication in example 1 (continued)

LDAP entry	Operation	Behavior
	modify-replace (userPassword)	Cannot perform a modify-replace of the userPassword attribute because the entry is subject to native authentication and password replace is not allowed.
cn=User2,ou=END,o=IBM,c=US	All	Entry is not configured for native authentication so all operations are regular LDAP operations.
cn=User3,ou=END,o=IBM,c=US	Bind	Attempts native authentication but fails because the Security Server ID USER3 is not defined, then a regular LDAP bind is performed.
	Bind with native password change	Cannot change the password on the bind because the Security Server ID USER3 is not defined.
	modify-delete and modify-add (userPassword)	Native password change is attempted but fails because the Security Server ID USER3 is not defined.
	modify-replace (userPassword)	An attempt to modify-replace the userPassword attribute fails because the entry is configured for native authentication.
cn=User4,ou=POK,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not contain the ibm-nativeId attribute.
cn=User5,ou=POK,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not contain the ibm-nativeId attribute.
cn=User6,ou=RAL,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not exist in a native subtree.

Example 2

- Assume these settings:

useNativeAuth all

nativeUpdateAllowed on

nativeAuthSubtree ou=END,o=IBM,c=US

nativeAuthSubtree ou=POK,o=IBM,c=US

the following table indicates the results of operations involving each user entry:

Table 16. Behavior of native authentication in example 2

LDAP Entry	Operation	Behavior
cn=User1,ou=END,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid ibm-nativeId .
	Bind with native password change	Can change this native password because the entry contains a valid ibm-nativeId .
	modify-delete and modify-add (userPassword)	Can change this native password because the entry contains a valid ibm-nativeId .
	modify-replace (userPassword)	Cannot perform a modify-replace of the userPassword attribute because the entry is subject to native authentication and password replace is not allowed.

Table 16. Behavior of native authentication in example 2 (continued)

LDAP Entry	Operation	Behavior
cn=User2,ou=END,o=IBM,c=US	Bind	Because there are no native attributes in this entry, a regular LDAP bind is attempted.
	Bind with native password change	Cannot change the password on the bind because the entry is not properly set up for native authentication. A regular LDAP bind is attempted.
	modify-delete and modify-add (userPassword)	Because there are no native attributes on this entry, native authentication password update is not attempted. A regular modification of the userPassword attribute value is attempted.
	modify-replace (userPassword)	Because there are no native attributes on this entry, native authentication password update is not attempted. A regular modification of the userPassword attribute value is attempted.
cn=User3,ou=END,o=IBM,c=US	Bind	Attempts native authentication but fails because the Security Server ID USER3 is not defined, then a regular LDAP bind is performed.
	Bind with native password change	Cannot change the native password on the bind because the Security Server ID USER3 is not defined.
	modify-delete and modify-add (userPassword)	Native password change is attempted but fails because the Security Server ID USER3 is not defined.
	modify-replace (userPassword)	An attempt to modify-replace the userPassword attribute fails because the entry is configured for native authentication.
cn=User4,ou=POK,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid uid (with one value).
	Bind with native password change	Can change this native password because the entry contains a valid uid (with one value).
	modify-delete and modify-add (userPassword)	Can change this native password because the entry contains a valid uid (with one value).
	modify-replace (userPassword)	An attempt to modify-replace the userPassword attribute fails because the entry is configured for native authentication.
cn=User5,ou=POK,o=IBM,c=US	Bind	Native bind fails because 2 uid values exist.
	Bind with native password change	Cannot change the native password on the bind because 2 uid attribute values exist.
	modify-delete and modify-add (userPassword)	Cannot change the native password on modify operations because 2 uid attribute values exist.
	modify-replace (userPassword)	An attempt to modify-replace the userPassword fails because the entry is configured for native authentication.

Table 16. Behavior of native authentication in example 2 (continued)

LDAP Entry	Operation	Behavior
cn=User6,ou=RAL,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not exist in a native subtree.

Using Native Authentication with Web Servers

Many Web servers provide a user ID and password challenge for authentication. These can take advantage of native authentication. The Web server must be configured to do LDAP authentication. When the challenge to do LDAP authentication is presented, the user can enter the Security Server user ID and password or password phrase (from the system where the LDAP server is running). The Web server will search the LDAP directory for an entry where **uid** equals the input user ID. The Web server will use the returned DN and the inputted password or password phrase to do an **ldap_simple_bind()**. When the LDAP server determines this entry is subject to native authentication, it will retrieve the **ibm-nativeId** or **uid** value and verify the password or password phrase with the Security Server. Note that if **useNativeAuth** is set to **selected**, it may be necessary to place the Security Server user ID into both the **uid** and **ibm-nativeId** attributes of this entry to allow the Web server processing to work correctly with native authentication.

Configuring Other Backends, SSL/TLS, Remote Services, and Encryption

If you plan to use:	See:
SDBM backend	“Setting up for SDBM” on page 111 , then return to “Step 7. Set the Time Zone” on page 105
GDBM backend	“Setting up for GDBM” on page 112 , then return to “Step 7. Set the Time Zone” on page 105
Remote services support (ICTX backend)	“Configuring remote services support” on page 114 , then return to “Step 7. Set the Time Zone” on page 105
SSL/TLS	“Setting up for SSL/TLS” on page 114 , then return to “Step 7. Set the Time Zone” on page 105
Encryption	“Configuring for Encryption or Hashing” on page 121 , then return to “Step 7. Set the Time Zone” on page 105
Plug-in extensions	“Configuring Plug-in Extensions” on page 125 , then return to “Step 7. Set the Time Zone” on page 105

CDBM Backend Configuration and Policy Entries

When the CDBM backend is configured in the LDAP server configuration file, configuration related entries are stored in the **cn=configuration** suffix while policy entries are stored in the **cn=ibmpolicies** suffix. These entries contain attributes that represent configuration options. The attribute values can be changed dynamically by an LDAP modify command while the LDAP server is running. All changes take effect immediately, without needing to restart the server. By default, the CDBM backend only allows the LDAP root administrator to modify the configuration and policy entries, but access can be changed by modifying the ACL on these entries. Some administrative roles allow modifying configuration and policy entries in the CDBM backend without modifying ACLs. See [Administrative group and roles in z/VM: TCP/IP LDAP Administration Guide](#) for more information.

When the LDAP server starts, the configuration and policy entries that do not exist are created with each attribute assigned to its default value. If an attribute value is deleted, the default value is used. The deleting and renaming of advanced replication configuration entries is supported only when **useAdvancedReplication off** is specified in the CDBM backend.

This section discusses the entries that exist under the **cn=configuration** and **cn=ibmpolicies** suffixes and the attribute values in these entries that affect the configuration of the LDAP server.

cn=configuration

This is a container entry that is used to define dynamic configuration attributes. [Table 17 on page 100](#) describes the entry attribute descriptions.

<i>Table 17. cn=configuration entry attribute descriptions</i>	
Attribute description and default	
cn	Specifies the common name of the configuration entry. This attribute is never interpreted by the server. Default: Configuration
ibm-slapdSAFSecurityDomain	Specifies the high-level component of the resource profile names that are used to define LDAP-related information in the z/VM security manager. This high-level qualifier is used to define administrative roles in the z/VM security manager. The length of this value cannot be more than 228 characters and cannot contain a blank, comma, parenthesis, semicolon, asterisk, percent sign, or ampersand. See Administrative group and roles in z/VM: TCP/IP LDAP Administration Guide for more information about configuring administrative roles in the z/VM security manager. Default: GLDSEC
ibm-slapdAdminGroupEnabled	A boolean (true or false) used to specify if the administrative group is currently enabled. If set to true, the administrative group is enabled and the LDAP root administrator can delegate server administration authority. If set to false, the administrative group cannot be enabled in the LDAP server. Default: false
ibm-slapdPagedResAllowNonAdmin	A boolean (true or false) used to indicate whether the server allows non administrators to request paged search results. If set to true, the server accepts any paged search request, including those submitted by a user binding anonymously. If set to false, the server only accepts paged search requests submitted by a user with administrator authority. In this case, the criticality of the pagedResults server control determines how the server handles a paged search request from non administrator users. If the control is specified as critical, the request is rejected with an LDAP_INSUFFICIENT_ACCESS return code. If the control is specified as non critical, the search is performed, but all results are returned without paging. The ibm-slapdPagedResLmt attribute must be set to a value greater than zero to enable paged search results. Default: false Note: This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.
ibm-slapdPagedResLmt	Specifies the maximum number of outstanding paged search requests allowed simultaneously on a single connection. If the maximum number of outstanding paged search requests is exceeded, the criticality of the pagedResults server control determines how the server handles the paged search request. If the control is specified as critical, the request is rejected with an LDAP_ADMIN_LIMIT_EXCEEDED return code. If the control is specified as non critical, the search is performed but all results are returned without paging. The value must be between 0 and the maximum integer size. A value of 0 indicates that paged search results are not supported. Default: 0 Note: This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.

<i>Table 17. cn=configuration entry attribute descriptions (continued)</i>
Attribute description and default
ibm-slapdServerID Specifies a short descriptive name of this server in an advanced replication environment. This server name is used when configuring the relationships among LDAP servers in an advanced replication environment and therefore a unique ibm-slapdServerID value is chosen for each server in the replication topology. This value is displayed as the ibm-serverID attribute value in the root DSE entry. The ibm-slapdServerID value is used in the replica subentry attribute ibm-replicaServerID; therefore, this value cannot be modified once replica subentries are created in the directory. This value cannot be deleted if advanced replication is configured. Default: A randomly generated attribute value like an ibm-entryUUID attribute value that is created when the CDBM backend is first initialized.
ibm-slapdSortKeyLimit Specifies the maximum number of sort keys that can be included on a single sorted search request. If the maximum number of sort keys is exceeded, the criticality of the SortKeyRequest server control determines how the server handles the sorted search request. If the control is specified as critical, the request is rejected with an LDAP_UNAVAILABLE_CRITICAL_EXTENSION return code. If the control is specified as non-critical, the search is performed, but unsorted results are returned and an LDAP_ADMIN_LIMIT_EXCEEDED sort result code is returned in the SortKeyResponse server control. The value must be between 0 and the maximum integer size. A value of 0 indicates that sorted search results are not supported. Default: 0 Note: This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.
ibm-slapdSortSrchAllowNonAdmin A boolean (true or false) used to indicate whether the server allows non-administrators to request sorted search results. If set to true, the server accepts any sorted search request, including those submitted by a user binding anonymously. If set to false, the server only processes sorted search requests submitted by a user with administrator authority. In this case, the criticality of the SortKeyRequest server control determines how the server handles a sorted search request from non-administrator users. If the control is specified as critical, the request is rejected with an LDAP_UNAVAILABLE_CRITICAL_EXTENSION return code. If the control is specified as non-critical, the search is performed but unsorted results are returned and an LDAP_INSUFFICIENT_ACCESS sort result code is returned in the SortKeyResponse server control. The ibm-slapdSortKeyLimit attribute must be set to a value greater than zero to enable sorted search results. Default: false Note: This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.

cn=Replication,cn=configuration

This entry is used to configure many aspects of advanced replication such as the maximum number of pending or failed replication changes displayed for a replication agreement. For more information about advanced replication, see [Advanced replication](#) in *z/VM: TCP/IP LDAP Administration Guide*.

<i>Table 18. cn=Replication,cn=configuration entry attribute descriptions</i>
Attribute description and default
cn Specifies the common name of the configuration entry. This attribute does not affect advanced replication configuration. Default: Replication
ibm-replicationOnHold A boolean (true or false) used to indicate whether replication is suspended from all replication agreements in the server. If set to true, replication from all replication agreements is suspended and updates are queued. If set to false, replication updates are handled normally by each replication agreement. Default: false
ibm-slapdMaxPendingChangesDisplayed Specifies the maximum number of pending replication changes and the maximum number of failed replication changes that are displayed when searching a replication agreement on a supplier server. Increase this value if more pending and failed changes must be displayed for each replication agreement. The pending replication changes are stored in the replication agreement entry in the ibm-replicationPendingChanges multi-valued operational attribute. The failed replication changes are stored in the ibm-replicationFailedChanges multi-valued operational attribute. For more information about these operational attributes, see the table on ibm-replicationAgreement operational attributes in Monitoring and diagnosing advanced replication problems in z/VM: TCP/IP LDAP Administration Guide. The value must be 0 - the maximum integer size. A value of 0 indicates that no pending changes are displayed for each replication agreement. Default: 200
ibm-slapdReplConflictMaxEntrySize Specifies the maximum length (in bytes) for all attribute values in an entry for replication conflict resolution to occur. If a replication conflict occurs on the consumer server and the total attribute value length for all values in an entry is less than or equal to this number, the entry is resent to the consumer server to automatically correct the replication conflict. Otherwise, the entry is not resent to the consumer server. This value applies to each replication agreement in the server. Increase this value when large entries are modified so that out of sync conditions between a supplier and consumer server can be resolved automatically with conflict resolution. If automatic replication conflict resolution support is not wanted, set this value to a small number. The value must be 0 - the maximum integer size. A value of 0 indicates that all entries are resent to the consumer server regardless of the size of the entry. Default: 0
ibm-slapdReplContextCacheSize Specifies the maximum size of each advanced replication context cache, in bytes. An advanced replication context cache is used to store pending replication updates for each replication context in the server. This cache reduces the number of queries to the backends to find the same information. Increase the size of the cache when replicating more and larger entries, such as large group entries. This value must be 0 - the maximum integer size. A value of 0 indicates that there are no replication context caches in the server. Default: 100000

<i>Table 18. cn=Replication,cn=configuration entry attribute descriptions (continued)</i>
Attribute description and default
ibm-slapdReplMaxErrors Specifies the maximum number of advanced replication failures that are logged for each backend in the server. If there are multiple replication agreement entries in a backend, each agreement shares the maximum number of replication failures allowed for the backend. The failed replication changes are stored in the replication agreement entry in the ibm-replicationFailedChanges multi-valued operational attribute. When the number of replication failures exceeds this value, advanced replication for this agreement stalls. For more information about recovering from out of sync and stall conditions, see Monitoring and diagnosing advanced replication problems in <i>z/VM: TCP/IP LDAP Administration Guide</i>. This value must be -1 - the maximum integer size. A value of 0 indicates that advanced replication failures are not logged for any replication agreements, therefore, replication stalls at the first failed replication update. A value of -1 indicates that an unlimited number of advanced replication failures are logged for all replication agreements. Default: 0
ibm-slapdReplRestrictedAccess A boolean (true or false) used to control access to replication topology entries (replication contexts, groups, subentries, and agreements). This attribute provides a way to limit access to the replication topology entries in the LDAP server. If set to true, only LDAP root, directory data, or replication administrators, and the master server DN have access to replication topology entries. If set to false, non-administrator users must have the proper authority to access the replication topology entries. Default: false Note: This is a user-modifiable operational attribute and is only returned on a search request when listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.
ibm-slapdReplicateSecurityAttributes Enables replication of security attributes between the read-only replica and master so that password policy for account lockout can be enforced in replication topologies. Note: This capability is only supported for bind operations that use simple authentication, and authentication that is done with compare operations that involve the userPassword attribute. Default: false

cn=Log Management,cn=Configuration

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=Replication,cn=Log Management,cn=Configuration

This entry is used by advanced replication to specify the location of the lost and found log file. For more information about advanced replication, see [Advanced replication](#) in *z/VM: TCP/IP LDAP Administration Guide*.

<i>Table 19. cn=Replication,cn=Log Management,cn=Configuration entry attribute descriptions</i>
Attribute description and default
cn Specifies the common name of the configuration entry. This attribute does not affect advanced replication configuration. Default: Replication

<i>Table 19. cn=Replication,cn=Log Management,cn=Configuration entry attribute descriptions (continued)</i>
Attribute description and default
ibm-slapdLog Specifies the filename and directory location for the lost and found log file. The lost and found log file is created by the consumer server the first time a replication conflict occurs. Any entries that are deleted because of a replication conflict are stored in LDIF format in this file. The directory path that is specified in this attribute value must exist before the file is created, otherwise replication conflicts are not written to the lost and found log file. This value cannot be deleted when advanced replication is configured. If this value is modified, the original value is still used until the LDAP server is restarted. Default: /var/ldap/logs/lostandfound.log

cn=adminingroup,cn=configuration

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=safadminingroup,cn=configuration

This entry is used to define administrative group members whose roles are defined in RACF. Group members are added to the entry by adding the optional member attribute to the entry and roles are defined in RACF. See [Administrative group and roles](#) in *z/VM: TCP/IP LDAP Administration Guide*.

<i>Table 20. cn=safadminingroup,cn=configuration entry attribute descriptions</i>
Attribute description and default
cn A required attribute that specifies the common name of the SAF administrative group entry. This attribute does not affect administrative role processing. Default: safadminingroup
member An optional attribute that specifies a distinguished name (DN) leading to a SAF user ID. Examples of these DNs are: SDBM entries, DNs from an SSL client certificate, and a DN of an LDBM entry participating in native authentication. This DN is checked against each administrative role defined in the LDAP general resource class in RACF to see if the user has READ authority to the profile. If the user has READ access to the profile, the user is granted that administrative role. Default: none

cn=ibmpolicies

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=pwdpolicy,cn=ibmpolicies

This entry is used to configure the global password policy. When the global password policy is activated, this policy applies to all entries that have passwords stored in the CDBM backend unless there is an overriding individual or group password policy in effect. For the attribute descriptions of the **cn=pwdpolicy,cn=ibmpolicies** entry, see [Password policy attributes](#) in *z/VM: TCP/IP LDAP Administration Guide*.

Step 7. Set the Time Zone

The LDAP server uses time values returned by the operating system when it records server activity or when it generates LDAP trace records. The LDAP server assumes that time values are in Universal Time Coordinated (UTC) format. The UTC time value is mapped to a (local) time zone value as specified by the **TZ** environment variable. By default, **TZ** is set to GMT0.

When started, the LDAP server will read an environment variable file. The default file is DS ENVVARS. This default can be changed by setting the environment variable **LDAP_DS_ENVVARS_FILE** to the full path name of the desired environment variable file. LDAP server timestamps are then generated using a UTC value and the time zone value.

1. If you have not already done so, copy DS ENVVARS to TCPMAINT 198.
2. Edit DS ENVVARS, uncomment the **TZ** environment variable and set the value as desired.

For more information about time zones, see the `tzset()` function description in [XL C/C++ for z/VM: Runtime Library Reference](#).

Step 8. Set Environment Variables (DS ENVVARS)

There are a number of environment variables that are processed by the LDAP server and utilities. Except for **LDAP_DS_ENVVARS_FILE**, they can be specified in the LDAP server environment variables file. By default, the file name is DS ENVVARS. The name can be reset using the **LDAP_DS_ENVVARS_FILE** environment variable. Environment variables are read once, during LDAP server initialization. The LDAP server must be stopped and restarted to put a change to an environment variable into effect.

Creating the DS ENVVARS file

A sample LDAP server environment variable file is provided as LDAPDS SAMPENVR on the TCPMAINT 591 disk. Copy your customized environment variable file to the TCPMAINT 198 minidisk as DS ENVVARS.

The list below describes the LDAP server environmental variables:

LDAP_CHANGELOG_DLL=dll_name

Used for processing change log requests from an external security manager (for example, RACF). The environment variable specifies the name of the external security manager DLL (`dll_name`) that LDAP loads at initialization. The default is RPICLM00. The external security manager DLL must provide two functions:

- A request processing function
- A termination function.

LDAP creates a thread that runs the request processing function in the external security manager DLL, passing a pointer to its change log request function as a parameter. This thread needs to initialize the external security manager support and then perform a thread-safe wait for change log requests from the external security manager. When a request is received, the LDAP change log request function should be called, passing a fullword-aligned parameter list with the following format:

```
LdapPcWord fullword,      ! Zeros
LdapPcInputLen fullword,  ! Length of BER request
LdapPcInput@ fullword     ! Pointer to BER request
```

`LdapPCWord` must be zeros, `LdapPcInput@` must contain the address of the BER encoded request, and `LdapPcInputLen` must contain the length of the request pointed to by `LdapPcInput@`.

GLDLOG_MICROSECONDS=ON | anything_else

Controls whether all generated log records contain microseconds in their timestamps. Microseconds are added to the timestamp if the value is set to **ON**. Microseconds are not included if the value is not **ON** or if the environment variable is not specified. See [“Activity logging” on page 163](#) for more information.

GLDLOG_MSG=MSGs | NOMSGs

Controls whether activity log records are generated when messages are created by the LDAP server. Messages are not written to the log if the environment variable is not specified. The **GLDLOG_MSG** environmental variable is deprecated. See [“Activity logging” on page 163](#) or the **logFileMsgs** configuration option at [“Configuration File Options” on page 132](#) for more information.

GLDLOG_OPS=WRITEOPS | ALLOPS | SUMMARY

Controls which operations generate LDAP server activity log records. No operations are logged if the environment variable is not specified. The **GLDLOG_OPS** environmental variable is deprecated. See [“Activity logging” on page 163](#) or the **logFileOps** configuration option at [“Configuration File Options” on page 132](#) for more information.

GLDLOG_TIME=TIME | NOTIME | MERGEDRECORD

Controls whether LDAP server activity log records are generated when the operation being logged ends. Log records are not generated when an operation ends if the environment variable is not specified or is set to NOTIME. The **GLDLOG_TIME** environmental variable is deprecated. See [“Activity logging” on page 163](#) or the **logFileRecordType** configuration option at [“Configuration File Options” on page 132](#) for more information.

IBMSLDAP_REPL_UPDATE_EXTRA_SECS=*interval*

Specifies, in seconds, how long the advanced replication engine waits for a replication operation to complete before setting an LDAP_TIMEOUT error code. By default, the advanced replication engine waits 60 seconds.

LDAP_ADVREPL_CLEANUP_INTERVAL=*interval*

Specifies, in seconds, how often backends participating in advanced replication delete replicated updates. If an incorrect value or 0 is specified, the value is set to 900 (delete replicated updates every 15 minutes). This value is also used if the environment variable is not specified.

LDAP_CONSOLE_LEVEL=I | W | E | A

Specifies the message severity level for sending a message created by the LDAP server to the operator console. Messages with a severity equal to or higher than the specified severity are sent to the operator console in addition to the normal output destination. Note that some LDAP server messages are always written to the operator console and are not affected by this value. Messages with a severity of E or higher are sent to the operator console if the environment variable is not specified.

LDAP_DEBUG=*level*

Specifies the desired debug level. The value for **LDAP_DEBUG** is a mask that you can specify in the following ways:

- A decimal value (for example, 32)
- A hexadecimal value (for example, x20 or X20)
- A keyword (for example, FILTER)
- A construct of these values using plus and minus signs to indicate inclusion or exclusion of a value.

For more information, see [“Debug Levels” on page 107](#).

LDAP_DEBUG_FILENAME=*filename*

Specifies the fully-qualified name of the LDAP trace output file. The trace output is written to stdout, if this environment variable is not specified. The trace file is not used if LDAP tracing is not active. If using an output file, make sure that the file is not being used for any other purpose.

The current process identifier is included as part of the trace file name when the name contains a percent sign (%).

Example: If **LDAP_DEBUG_FILENAME** is set to /tmp/ldap.%.trc and the current process identifier is 247, the trace file name is /tmp/ldap.247.trc.

LDAP_DS_ENVVARS_FILE=*filename*

Specifies the name of the name of the LDAP server environment variables file. If the environment variable is not specified, the file name is DS ENVVARS.

LDAP_ERROR_LOGGING=STDOUT | STDERR | BOTH

Specifies how error messages are logged. The following values can be specified:

STDOUT

Error messages are written to standard output as specified by the LDAP_STDOUT_FILENAME environment variable.

STDERR

Error messages are written to standard error as specified by the LDAP_STDERR_FILENAME environment variable.

BOTH

Error messages are written to both standard output and to standard error.

Error messages are written to standard error, if this environment variable is not specified.

LDAP_NETWORK_POLL *interval*

Specifies, in minutes, how often the LDAP server will poll a network interface to determine if it has failed or has become active. If 0 is specified, the value is set to 5 (poll every 300 seconds). This is also the value used if the environment variable is not specified.

LDAP_PRINT_CONFIG=1 | *anything_else*

Controls whether the configuration options used by the LDAP server are displayed when the LDAP server is started. The options are displayed if the value is 1 and are not displayed for any other value. The options are displayed if the environment variable is not specified.

LDBM_SHUTDOWN_FAST=1 | *anything_else*

Controls how an LDBM backend in the LDAP server stops. Server shutdown normally frees all storage allocated and held by each backend. This freeing can potentially include a large amount of storage for LDBM because it holds all its entries in memory, and therefore can be very time consuming. When the value is set to 1, storage allocated by the LDBM backend is not released before the LDAP server stops (the storage is eventually released by the system). When the value is not 1 or if the environment variable is not specified, the storage is freed before the LDAP server stops.

LDAP_STDERR_FILENAME=*filename*

Specifies the fully-qualified name of the file to receive standard error messages generated using LDAP message services. If this environment variable is not specified, messages are written to stderr. Make sure that the output file is not being used for any other purpose.

LDAP_STDOUT_FILENAME=*filename*

Specifies the fully-qualified name of the file to receive standard output messages generated using LDAP message services. If this environment variable is not specified, messages are written to stdout. Make sure that the output file is not being used for any other purpose.

LDAP_USE_INTERNAL_RNG=0 | *anything_else*

Specifies which random byte generator the LDAP server uses for the creation of the salt value for Salted SHA (SSHA) hashing and the generation of random data bytes for CRAM-MD5 and DIGEST-MD5 authentication binds. If the value is 0 or the environment variable is not specified, the LDAP server uses the SSL random byte generator provided in the **gsk_generate_random_bytes()** routine. When the value is not 0, the LDAP server uses an internal random byte generator.

Debug Levels

The debug level is a mask that may be specified as follows:

- A decimal value (for example, 32)
- A hexadecimal value (for example, x20 or X20)
- A keyword (for example, FILTER)
- A construct of those values using plus and minus signs to indicate inclusion or exclusion of a value. For example:
 - '32768+8' is the same as specifying 'x8000+x8', or 'ERROR+CONNS'
 - '2147483647-16' is the same as specifying 'x7FFFFFFF-x10' or 'ANY-BER'
 - By beginning the debug level with a minus sign, you can deactivate debug collection for a debug level. For example, "-CONNS" modifies an existing debug level by deactivating connection traces.

- By beginning the debug level with a plus sign, you can activate debug collection for a debug level. For example, "+CONNS" modifies an existing debug level by activating connection traces.

Note: Specifying the debug level using decimal or hex values with a plus or minus sign is not necessarily the same as specifying the sum or difference as the debug level. For example, specifying '7+1' activates the 'TRACE', 'PACKETS', and 'ARGS' debug levels, while specifying '8' activates only the 'CONNS' debug level. Similarly, specifying '16-1' activates only the 'BER' debug level, while specifying '15' activates 'TRACE', 'PACKETS', 'ARGS', and 'CONNS'.

Table 21 on page 108 lists the debug levels and the related decimal, hexadecimal, and keyword values.

Table 21. Debug levels

Keyword	Decimal	Hexadecimal	Description
OFF	0	0x00000000	No debugging
TRACe	1	0x00000001	Entry and exit from routines
PACKets	2	0x00000002	Packet activity
ARGS	4	0x00000004	Data arguments from requests
CONNs	8	0x00000008	Connection activity
BER	16	0x00000010	Encoding and decoding of data, including ASCII and EBCDIC translations, if applicable
FILTer	32	0x00000020	Search filters
MESSage	64	0x00000040	Messaging subsystem activities and events
ACL	128	0x00000080	Access Control List activities
STATs	256	0x00000100	Operational statistics
THREad	512	0x00000200	Threading activities
REPLication	1024	0x00000400	Replication activities
PARSe	2048	0x00000800	Parsing activities
PERFormance	4096	0x00001000	Performance statistics
SDBM	8192	0x00002000	Backend activities (SDBM)
REFErral	16384	0x00004000	Referral activities
ERROr	32768	0x00008000	Error conditions
MULTIServer	131072	0x00020000	Multi-server activities
LDAPBE	262144	0x00040000	Connection between a frontend and a backend
STRBuf	524288	0x00080000	UTF-8 support activities
TDBM	1048576	0x00100000	Backend activities—TDBM (backend not supported)
SCHEma	2097152	0x00200000	Schema support activities
BECApabilities	4194304	0x00400000	Backend capabilities
CACHe	8388608	0x00800000	Cache activities
INFO	16777216	0x01000000	General processing information
LDBM	33554432	0x02000000	Backend activities (LDBM)
ANY	2147483647	0x7FFFFFFF	All levels of debug

Table 21. Debug levels (continued)

Keyword	Decimal	Hexadecimal	Description
ALL	2147483647	0x7FFFFFFF	All levels of debug

Note: The minimum abbreviation for each keyword is shown in uppercase letters.

The debug level for the server can be set at a number of different times.

- The initial debug level is OFF.
- Prior to starting the server, the LDAP_DEBUG environment variable may be set. The server uses this value first. For example,

```
export LDAP_DEBUG='ERROR+TRACE'
```

- When starting the server, the **DEBUG** parameter may be specified. The debug level specified on this parameter replaces the preceding debug level.
- It is possible to change the debug level while the server is running. See [“Dynamic Debugging” on page 163](#) for more information.

Step 9. Verify the LDAP Server

The following examples show how you can verify your LDAP server using LDAPSrch (the ldapsearch utility).

Note:

1. **Verify LDBM.** In the command below, substitute the **suffix** value from your LDAP server configuration file for the -b parameter. The command can be run multiple times to verify each suffix defined in the configuration file.

```
ldapsrch -h 127.0.0.1 -s base -b "o=Your Company" "objectclass=*
```

Note:

- a. If **allowAnonymousBinds off** is specified in the LDAP server configuration file, you must specify a distinguished name to bind with using the **-D** and **-w** options on LDAPSrch.
 - b. The LDAP search returns the message "No such object" if the suffix entries have not been loaded into the directory. The LDBM suffix entries can be added by using the steps outlined in [“Step 10. Finalize Setup of LDAP Backends” on page 110](#) and then this LDAP search can be tried again to verify that the entry is correctly added.
2. **Verify SDBM.** For SDBM, you must bind with a valid RACF-style DN to perform the search. Substitute a RACF ID of your choice in the racfid portion of the DN on the -D and the -b parameters below. Also, substitute your SDBM suffix in the DN on the -D and -b parameters. The RACF password for the user ID used in the -D parameter must be specified in the -w parameter.

```
ldapsrch -h 127.0.0.1 -D racfid=IBMUSER,profiletype=user,cn=myRacf
-w password_for_IBMUSER -b racfid=IBMUSER,profiletype=user,cn=myRacf
"objectclass=*
```

3. **Verify GDBM.** For GDBM, you must bind with the LDAP root administrator DN or another DN authorized to search the change log.

```
ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=changelog
"objectclass=*
```

4. **Verify CDBM.** For CDBM, you must bind with the LDAP root administrator DN or another DN authorized to search the cn=ibmpolicies and cn=configuration CDBM suffixes.

```
ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=ibmpolicies
"objectclass=*
```

```
ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=configuration
"objectclass=*
```

The previous LDAPSrch examples assume a default port of 389. If your port is not 389, use the `-p` parameter to specify the correct port.

Be sure to substitute the correct TCP/IP host name or TCP/IP address for the 127.0.0.1 after the `-h` parameter. The `-b` parameter specifies the starting point for the search. The use of the quotation marks around the filter prevents the asterisk (*) from being interpreted by the shell.

For more information about LDAPSrch, see [z/VM: TCP/IP User's Guide](#).

Step 10. Finalize Setup of LDAP Backends

To finalize setup of LDAP backends, follow these steps:

1. If you have configured an LDBM backend, modify the LDAP server schema entry to contain the schema needed for your usage of the backend. The USRSCHM LDIF and IBMSCHM LDIF on the TCPMAINT 591 disk might contain the schema you need. IBMSCHM LDIF requires that you first load USRSCHM LDIF. Additional schema might also be needed by the applications that are going to use the LDBM directory. For more information, see [LDAP directory schema](#) in [z/VM: TCP/IP LDAP Administration Guide](#).

Note:

- a. The distinguished name (DN) of the LDAP server schema is `cn=schema`. If the `ldif` file containing your schema has a DN of `cn=schema, suffix`, then update the file to change the DN to `cn=schema`.
 - b. Do not add any additional schema to use the GDBM, CDBM, or SDBM backends unless you are using SDBM to access RACF customized fields or adding user-defined entries to CDBM. The initial schema built into the LDAP server is typically sufficient for these backends.
2. Load the suffix entries for each configured LDBM backend. Do not try to load a suffix entry for the GDBM, CDBM, or SDBM backend. The GDBM, CDBM, and SDBM suffix entries are generated automatically by the LDAP server.

Use the LDAPADD utility to load the suffix entry.

```
ldapadd -h ldaphost -p ldapport -D binddn -w passwd -f file
```

where *file* is a file containing the entries to be loaded in LDIF format. For example, if `suffix.ldif` contains the following suffix entry:

```
dn: o=Your Company
objectclass: organization
o: Your Company
```

then the suffix entry can be added by LDAPADD utility as follows:

```
ldapadd -h myhost -p 389 -D cn=admin -w secret -f suffix.ldif
```

For more information about the LDAPADD utility, see [z/VM: TCP/IP User's Guide](#).

3. Load additional entries into the LDBM or CDBM backend, as you want. You can use the LDAPADD utility to load entries into these backends. You must load entries in hierarchical order and an entry cannot be loaded before its parent entry is loaded.

You can also add entries to SDBM (using LDAPADD), but this results in adding profiles to RACF. For more information, see [Accessing RACF information](#) in [z/VM: TCP/IP LDAP Administration Guide](#).

You cannot load entries into the GDBM backend. GDBM entries are created only by the LDAP server itself.

4. If GDBM is configured, set an appropriate ACL for controlling access to change log entries in the GDBM backend. For more information, see [Change logging](#) in [z/VM: TCP/IP LDAP Administration Guide](#).

5. If CDBM is configured, set an appropriate ACL for controlling access to the **cn=ibmpolicies** suffix because by default all users have access to that suffix. The ACL set on the **cn=configuration** suffix only allows the LDAP administrator authority to that suffix. For more information, see [Password policy in z/VM: TCP/IP LDAP Administration Guide](#).
6. After initial set-up of the LDAP server, you might want to remove the **adminPW** configuration option from the LDAP server configuration file. For more information, see [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#).

Setting up for SDBM

The LDAP server can provide remote LDAP access to the user, group, connection, and general resource profile information stored in RACF. It also supports setting RACF options that affect classes. For details about how you can use this RACF information, see [Accessing RACF information in z/VM: TCP/IP LDAP Administration Guide](#). When creating change log records for changes to RACF data, SDBM is required.

In order for your LDAP server to have access to RACF data, you must have RACF Security Server for z/VM, function level 530 (or later) installed on your system. For information on installing and configuring RACF, see [RACF Program Directory](#).

In order to configure your LDAP server to run with the SDBM backend of the LDAP server:

1. If you have not already done this, copy the configuration files from the TCPMAINT 591 disk to the TCPMAINT 198 disk (see [“Step 5. Copy the Configuration Files” on page 83](#)).
2. Use the following lines in your DS CONF file:

```
database sdbm GLDBSD31
suffix "your_suffix"
```

where *your_suffix* is any valid DN (distinguished name). Be sure to provide a meaningful value for the suffix. For example, a valid suffix line is:

```
suffix "cn=RACFA,o=IBM,c=US"
```

3. For RACF to be able to log changes to a RACF user, group, or connection:
 - Configure the SDBM backend. The SDBM suffix is needed to create a DN for the change log entry for a modification to a RACF user, group, or connection. SDBM is also needed to retrieve the RACF user's new password or other changed fields.
 - Enable the LDAP Program Call support in the LDAP server containing the change log. To do this, add the following option to either the global section of the configuration file or to the -l command-line parameter when starting the LDAP server (see [“LDAPSRV Command Syntax” on page 80](#)):

```
listen ldap://:pc
```

Note: This listen parameter for LDAP Program Call support is an addition to any other listen parameters you have specified. For example:

```
listen ldap://:pc
listen ldap://:389
```

- Add the following statement to the CP directory entry for the LDAP server user ID:

```
IUCV *RPI PRIORITY MSGLIMIT 255
```

Note:

1. Only one SDBM backend can be defined in any given LDAP server.
2. The attributes and object classes used by SDBM are always in the LDAP server schema.
3. The **enableResources** configuration option must be specified in your DS CONF file if you intend to display or manage RACF resource profiles and class options. This configuration option is also required

if you want to create change log entries for changes to RACF resource profiles. For more information, see [“Configuration File Options”](#) on page 132.

For other SDBM-specific information, see [Accessing RACF information in z/VM: TCP/IP LDAP Administration Guide](#).

Setting up for GDBM

The LDAP server can provide a change log containing information about changes to:

- RACF users, groups, and user-group connections, and general resource profiles
- LDBM and CDBM entries
- LDAP server schema entry.

GDBM can be configured to store change log entries in the Byte File System.

Note:

1. Only one GDBM backend can be defined in any given LDAP server.
2. The attributes and object classes used by GDBM are always in the LDAP server schema.
3. For additional configuration options that can be specified, see [Change logging in z/VM: TCP/IP LDAP Administration Guide](#).
4. If you intend to create change log entries for changes to RACF data, you must also configure an SDBM backend and enable the LDAP Program Call support. See [“Setting up for SDBM”](#) on page 111.

The amount of space needed to store a GDBM backend in an Byte File System depends on how many change log entries are going to be stored and the size of the change log entries. The number of change log entries can be controlled using the **changeLogMaxEntries** and **changeLogMaxAge** options in the GDBM section of the LDAP server configuration file. The size of a change log entry is related to the size of the LDIF when adding or modifying an LDBM or CDBM entry, because this LDIF is inserted into the change log entry. Generally, the space required to hold the GDBM backend data is:

```
6 X (maximum number of GDBM entries) X (largest add or modify LDIF + 1000)
```

This includes the extra space needed to copy the database files during GDBM commit processing.

When the LDAP server starts for the first time with a new GDBM backend configured, the server automatically creates the directories specified in the **databaseDirectory** server configuration option (or in **/var/ldap/gdbm** if the configuration option is not specified). When the directories are created, the LDAP server's userid is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP server's userid. The group that the LDAP server's userid belongs to is granted read and write access to the directories. As part of the GDBM backend initialization process, the server creates an **LDBM-1.db** files for the changelog backend, along with an **LDBM.ckpt** file. These files are created with the LDAP server's userid as the owner. The default permissions on these files grant read and write access to both the LDAP server's userid and the group to which the LDAP server's userid belongs.

If the default GDBM backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information on the **chmod** and **chown** commands, see [z/VM: OpenExtensions Commands Reference](#).

In order to configure your LDAP server to run with the GDBM backend of the LDAP server:

1. If you have not already done this, copy the configuration files from the TCPMAINT 591 disk (see [“Step 5. Copy the Configuration Files”](#) on page 83).
2. You need to use the following lines in your DS CONF file:

```
database gdbm GLDBGD31
```

The default file directory used by the GDBM backend to store its entries is `/var/ldap/gdbm`. If you do not want GDBM to use that file directory, then add the **databaseDirectory** option to the GDBM section of the configuration file. The file directory must be different than the ones used by any LDBM or CDBM backends. The LDAP server must have read-write access to the file directory. See [“Step 4. Set Up the User ID and Security for the LDAP Server”](#) on page 82.

Note: The files contained in the directory specified by the **databaseDirectory** server configuration option are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences could occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

Setting up for CDBM

The LDAP server provides the CDBM backend to store configuration information, for example, for advanced replication and password policy. CDBM is file-based, storing its directory information in the Byte File System.

The amount of space needed to store a CDBM backend in a file system is approximately four to six times the size of the expected input LDIF data. Generally, the space required to hold the CDBM backend data is two to three times the size of the expected input LDIF data. However, during the CDBM commit process each of the CDBM backend files is copied, therefore, resulting in occasionally needing twice the amount of file system space.

CDBM keeps its directory in storage in the LDAP virtual machine while the LDAP server is running.

When the LDAP server starts for the first time with the CDBM backend configured, the server automatically creates the directories specified in the **databaseDirectory** server configuration option (or in the directory specified by the **schemaPath** configuration option or in `/var/ldap/schema` if **schemaPath** is not specified). When the directories are created, the LDAP server's userid is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP server's userid. The group to which the LDAP server's userid belongs is granted read access to the directories. As part of the CDBM backend initialization process, the server creates **LDBM-1.db** and **LDBM-2.db** files for the **cn=configuration** and **cn=ibmpolicies** suffixes, along with an **LDBM.ckpt** file. These files are created with the LDAP server's userid as the owner. The default permissions on these files grant read and write access to the LDAP server's userid while the group to which the LDAP server's userid belongs is granted read access.

If the default CDBM backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information about the **chmod** and **chown** commands, see [z/VM: OpenExtensions Commands Reference](#).

In order to configure your LDAP server to run with the CDBM backend of the LDAP server:

- If you have not already done so, copy your customized LDAP configuration file to the TCPMAINT 198 minidisk as DS CONF.
- Use the following line in your DS CONF file:

```
database cdbm GLDBC31
```

Notes:

1. Only one CDBM backend can be configured in an LDAP server. If the **databaseDirectory** option is not specified in the CDBM backend section, the CDBM backend stores its entries in the directory specified by the **schemaPath** configuration option. If the **schemaPath** configuration option is not specified, the CDBM backend data is stored in the default **schemaPath** directory, which is `/var/ldap/schema`. The **databaseDirectory** value must be different than the ones used by any other LDBM or GDBM backend. The LDAP server must have read-write access to the file directory. See [“Step 4. Set Up the User ID and Security for the LDAP Server”](#) on page 82.
2. Depending on the types of entries you intend to add to the CDBM backend, you may need to add schema to the LDAP server schema for the attributes and objectclasses that are to be used. For

information about adding schema to the LDAP server, see [Setting up the schema for LDBM and CDBM in z/VM: TCP/IP LDAP Administration Guide](#).

3. The files contained in the directory specified by the **databaseDirectory** or **schemaPath** server configuration options are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences could occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

Configuring remote services support

Remote services allows programs that do not reside on z/VM or programs that reside on a z/VM guest to centralize authorization decisions and security event logging by using RACF Security Server for z/VM. These services are provided by the ICTX plug-in of the z/VM LDAP server.

For remote authorization calls to z/VM, the request comes in the form of a DER-encoding of the ASN.1 syntax. For details on these types of remote authorization and auditing requests, see [Remote authorization and auditing through LDAP in z/VM: TCP/IP Programmer's Reference](#).

To enable the database functions for remote services, the LDAP configuration file must have a section that identifies the ICTX extended operations support.

To configure the ICTX plug-in, add the following plugin configuration line to the LDAP DS CONF file. It must be specified before any database configuration sections.

```
plugin clientOperation GLDBIC31 ICTX_INIT "CN=ICTX"
```

Setting up for SSL/TLS

SSL support is provided with LDAP and does not need to be separately installed, nor does the LDAP server use the SSL/TLS services provided by the z/VM SSL server (you do not protect the LDAP server ports in the same manner as that currently used for other servers such as FTP). The LDAP server contains the ability to protect LDAP access with Secure Sockets Layer (SSL) and Transport Layer (TLS) security. There are two types of connections that support secure communication:

- An SSL/TLS only secure connection. This connection requires that the first communication between the client and the server be the handshake that negotiates the secure communication. From that point on only secure communication can occur on the connection.
- A bimodal connection that supports secure and non-secure communication. The client is expected to begin communication in a non-secure mode. At some time during communication, the client may change to secure communication by sending a **StartTLS** extended operation after which the handshake to negotiate secure communication occurs followed by secure communication. The client may shutdown secure communication causing a **StopTLS** alert to be sent and the server will continue communication in a non-secure mode. At a later time, the client may restart secure communication by sending another **StartTLS** extended operation followed by the handshake.

Both types of connections require that the SSL/TLS be configured for use by the LDAP server.

Using SSL/TLS Protected Communications

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols use public-key infrastructure (PKI) algorithms to establish and maintain an encrypted communications path between a client and server. In z/VM, the ability to set up and communicate over SSL/TLS protected communication links is provided by the LDAP server.

In order for the LDAP client to communicate with an LDAP server over an SSL/TLS-protected TCP/IP socket connection, the LDAP server must transmit a certificate to the LDAP client and, optionally, the client can transmit its certificate to the LDAP server. The LDAP client and server must verify that the certificates they received are valid. Once the LDAP client and server have determined the validity of the

certificates provided to them, SSL/TLS-protected communication occurs between the LDAP client and server.

The LDAP client and server verify the certificates sent to them by using public-key digital signatures. The LDAP client and server take the certificates and compare the digital signature in the certificates with a signature that it computes based on having the public-key of the signer of the certificate. In order to do this, the LDAP client and server must have the public-key of the signer of the certificates. The LDAP client and server obtain this by reading a file that contains these public-keys. This file is called a key database.

A key database contains the public-keys that are associated with signers of certificates. These public-keys are, in reality, contained in certificates themselves. Thus, verifying one certificate requires the use of a different certificate, the signer's certificate. In this fashion, a chain of certificates is established, with one certificate being verified by using another certificate and that certificate being verified by yet another certificate, and so on. A certificate, and its associated public key, can be defined as a *root* certificate. A root certificate is *self-signed*, meaning that the public-key contained in the certificate is used to sign the certificate. Using a root certificate implies that the user *trusts* the root certificate.

The key databases used by the LDAP client and server must contain enough certificates in order to verify the certificates sent by the LDAP client and server during the start up of the SSL/TLS connection. If either certificate is self-signed, then that certificate must be stored in the other's key database. If the certificates are signed by some other certificate signer, then the signer's certificate and any certificates that this certificate depends upon must be stored in the key databases. The key databases used by the LDAP client and server must also contain the certificates that they will transmit to each other during the startup of the SSL/TLS-protected communications.

Enabling the LDAP Server to Use IBM Z Cryptographic Hardware

There are two forms of cryptographic hardware available to IBM Z. The first is the CP-Assisted Cryptographic Facility (CPACF), which is a no-charge feature available on all modern generations of IBM Z and LinuxONE. This feature will accelerate use of symmetric algorithms (such as AES and 3DES) and hashes (such as SHA-1 and SHA-256). It is enabled at the partition level. LDAP for z/VM will make automatic use of this feature if available, with no configuration required.

The System SSL library supporting LDAP for z/VM can also offload cryptographic operations to the Crypto Express adapters. These adapters will accelerate asymmetric algorithms (such as clear-key RSA) in addition to the symmetric operations listed previously. In order to enable the LDAP for z/VM server to use this function, check the following steps. Note that these steps are required only if LDAP has been configured to use System SSL directly, rather than the TLS/SSL Server.

1. Verify that Crypto Express queues are available to your z/VM partition. This can be validated by issuing QUERY CRYPTO DOMAINS USERS from an authorized virtual machine. For more information about the QUERY CRYPTO command, see [z/VM: CP Commands and Utilities Reference](#). If cryptographic domains are available for shared usage, continue to Step 2. If no domains are available, validate your hardware configuration by checking the LPAR Activation Profile on the Hardware Management Console.
2. Insert a statement into the LDAPSRV virtual machine directory entry: 'CRYPTO APVIRTUAL'. This statement will grant the LDAP Server access to shared crypto domains associated with your z/VM partition.
3. If necessary, refresh your object directory by using DIRECTXA or appropriate Directory Manager commands.
4. Restart your virtual machine in order to establish access with the virtualized cryptographic hardware.
5. When your LDAP Server is running, logged on or disconnected, it will appear in the list of Shared Crypto users available via the QUERY CRYPTO command.

In the unlikely case that cryptographic hardware becomes unavailable to the virtual machine, the LDAP server will automatically fall back to software encryption mechanisms available in the System SSL cryptographic library.

Creating and Using a Key Database

The LDAP client and server use the SSL functions to set up SSL/TLS protected communications. The SSL capability requires a key database be set up before SSL/TLS protected communications can begin.

The key database is a password protected file stored in the Byte File System. This file is created and managed using a utility program called GSKKYMANT. For information about using the GSKKYMANT utility, see [SSL Certificate Management](#) in *z/VM: TCP/IP User's Guide*.

The key database file that is created must be accessible by the LDAP server.

For testing purposes, the LDAP server can use a self-signed certificate. In this case, the certificate of the LDAP server must also be stored in the key database of the LDAP client in order for SSL/TLS protected LDAP communications to work between the client and server.

If the SSL certificate that the LDAP server is going to use is not marked as the default certificate in the key database, then the **sslCertificate** server configuration option must be specified in order to determine which SSL certificate to use.

Obtaining a Certificate

The LDAP server or client can obtain a certificate by contacting a certificate authority (CA) and requesting a certificate. Utilities to formulate a certificate request are provided by GSKKYMANT. This certificate request is usually passed to the CA by means of an electronic mail message or by an HTML form which is filled out using a web browser. Once the CA verifies the information for the LDAP client or server, a certificate is returned to the requester, usually by an electronic mail message. The contents of the mail message are used to define the certificate in the key database.

Enabling SSL/TLS Support

The following high-level steps are required to enable SSL/TLS support for LDAP. These steps assume you have already installed and configured the LDAP server.

1. Generate the LDAP server private key and server certificate and mark it as the default in the key database or use its label on the **sslCertificate** option in the LDAP server configuration file.
2. Configure the LDAP server to the security options you want that are related to SSL/TLS secure communications. (see [“Setting up the Security Options for the LDAP Server”](#) on page 116).
 - Defining the acceptable SSL and TLS protocol levels.
 - Defining the acceptable cipher specifications.
 - Defining the secure sockets and bimodal sockets the server uses to listen for inbound client requests.
 - Defining the type of authentication wanted.
 - Defining the server certificate to be used, and where it is located.
3. Restart the LDAP server.

Setting up the Security Options for the LDAP Server

Several options that are related to the LDAP server SSL/TLS secure communications can be controlled by environment variables that are used by System SSL. The accepted protocol levels, the cipher suites, and suite B profile are all configured by using environment variables.

The z/VM LDAP server does not support SSL 2, and disables it from being used. SSL 3, TLS 1.0, TLS 1.1, and TLS 1.2 are supported. The System SSL defaults and environment variables control which of these supported protocols are enabled or disabled. For example, the **GSK_PROTOCOL_SSLV3** environment variable can be set to ON to enable SSL 3 or OFF to disable SSL 3. The **GSK_PROTOCOL_TLSV1** environment variable can be set to ON to enable TLS 1.0 or OFF to disable TLS 1.0. TLS 1.1 and TLS 1.2 are disabled by default. To enable TLS 1.1, set the **GSK_PROTOCOL_TLSV1_1** environment variable to ON. Similarly, to enable TLS 1.2, set the **GSK_PROTOCOL_TLSV1_2** environment variable to ON.

The LDAP server provides the **sslCipherSpecs** configuration option for specifying the accepted cipher suites that are used in SSL/TLS secure connections. However, this configuration option is limited in its support, and is provided only for compatibility with earlier versions. During the SSL handshake that occurs when a secure connection is established, System SSL uses the list of acceptable cipher specifications that are provided by the LDAP server to determine preference. When the **sslCipherSpecs** configuration option is used to specify the acceptable cipher suites, the order of preference is defined by the LDAP server and matches the default order that is documented by System SSL. The **sslCipherSpecs** specification as a bit-mask does not readily accommodate the many available cipher suites that are provided with TLS 1.1 and TLS 1.2, nor does it provide the capability of altering the default preference order.

If the only cipher specifications required are listed in [Table 22 on page 118](#), and the default order of preference is acceptable, the **sslCipherSpecs** configuration option can be used to list the cipher suites you want. Otherwise, you must specify "sslCipherSpecs GSK_V3_CIPHER_SPECS_EXPANDED" in your LDAP server configuration file. With this setting, the cipher suites that are accepted come from the defaults as defined by System SSL, including those specified with the GSK_V3_CIPHER_SPECS_EXPANDED environment variable. Set the GSK_V3_CIPHER_SPECS_EXPANDED environment variable to the 4-character cipher suites you want, in order of preference.

To restrict the SSL/TLS secure connections that are used by the LDAP server to the Suite B Profile, you must set the GSK_SUITE_B_PROFILE environment variable to the appropriate setting. See *z/OS Cryptographic Services System SSL Programming* for more information. If the Suite B profile is enabled, this controls the acceptable SSL/TLS protocol levels and cipher suites, overriding the other settings that are mentioned above.

The following options for SSL/TLS can be set in the LDAP server configuration file. They are described in detail in ["Configuration File Options" on page 132](#).

- listen
- sslAuth
- sslCertificate
- sslCipherSpecs
- sslKeyRingFile
- sslKeyRingFilePW
- sslKeyRingPWStashFile

Note:

1. The **replKeyRingFile** and **replKeyRingPW** options are no longer necessary or recognized by the LDAP server. These options should be removed from the configuration file.
2. The security, port, and **securePort** options are deprecated by the listen option. For more information, see [listen](#).

LDAP can be configured for SSL/TLS in two ways:

- For secure-only communication, specify one or more **listen** options for secure communications in the following format:

```
ldaps://[IP_address | hostname | INADDR_ANY | in6addr_any][:portNumber]
```

- For bimodal (non-secure/secure) communication, specify one or more **listen** options for non-secure communications in the following format:

```
ldap://[IP_address | hostname | INADDR_ANY | in6addr_any][:portNumber]
```

For more information on the **listen** option, see [listen](#).

The **sslKeyRingFile** option specifies the name of the key database used by the LDAP server. This key database is also used for SSL/TLS protected replication. Because the replicating server may be acting as both a replica server and an LDAP server, the replica server's certificate (or CA's certificate) must be contained in the replicating server's key database file.

A key database requires a password. The password may be specified on the **sslKeyRingFilePW** option or the name of a password stash file may be specified on the **sslKeyRingPWStashFile** option in the LDAP server configuration file. Use of a stash file provides a method of specifying a password in a form that can not be easily read by a human. The GSKKMAN utility provides a function to create the key database password stash file.

The LDAP server is configured to provide server and, optionally, client authentication. The **sslAuth** option controls this setting.

When using server authentication, by setting the **sslAuth** server configuration option to **serverAuth**, the LDAP server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the LDAP server to the LDAP client application. The LDAP server supplies the client with the LDAP servers X.509 certificate during the initial SSL handshake. If the client validates the servers certificate, then a secure, encrypted communication channel is established between the LDAP server and the LDAP client application.

In addition, if the LDAP server is configured to use server and client authentication, by setting the **sslAuth** server configuration option to **serverClientAuth**, and the client sends a digital certificate on the initial SSL handshake, it must be validated by the LDAP server before the secure encrypted communication channel is established between them. The certificate is used to establish the bind identity of the client.

Note: If the LDAP server is configured for both server and client authentication, but a client does not send a digital certificate, then the server will act as if configured for server authentication only. This is for compatibility with earlier versions of the LDAP server. In addition, System SSL can be configured to fail the SSL handshake if the client does not provide a certificate after the server requests client authentication. System SSL provides an environment variable, **GSK_CLIENT_AUTH_NOCERT_A**, which indicates that a client certificate must be passed to the server to prevent the SSL handshake from failing. See *z/OS Cryptographic Services System SSL Programming* for more information.

The **sslCertificate** option indicates the label of the server certificate that is to be used. This option is needed if the default certificate has not been set in the key database, or if a certificate other than the default certificate is desired.

The **sslCipherSpecs** option specifies the cipher specifications that are accepted from clients. This option is discouraged and provided only for compatibility with an earlier version. It supports only a portion of the cipher suites available in System SSL, contains no 4-character cipher suites, and provides no order of preference. The preferred approach is to specify "sslCipherSpecs GSK_V3_CIPHER_SPECS_EXPANDED" and use the GSK_V3_CIPHER_SPECS_EXPANDED environment variable to define the list of 4-character cipher specifications, in order of preference.

If the cipher specifications you want are included in [Table 22 on page 118](#), and if the order of preference matches the default order that is provided by System SSL, then the **sslCipherSpecs** option may be used with any of the values described.

If **sslCipherSpecs** is omitted from the server configuration file, all cipher specifications that are listed in [Table 22 on page 118](#) are used.

Depending upon the level of System SSL support, the list of acceptable cipher specifications might be lowered because certain specifications might not be supported by System SSL for that level of the product. [Table 22 on page 118](#) lists the LDAP sslCipherSpecs mask values that are accepted by the **sslCipherSpecs** option, and the related decimal, hexadecimal, and keyword values. It also lists the System SSL cipher number.

Table 22. SSL ciphers supported by the sslCipherSpecs configuration option

Cipher	Decimal value	Hexadecimal value	SSL value	Description
ALL	4294967295	FFFFFFFF		All cipher suites
ANY	4294967295	FFFFFFFF		All cipher suites

Table 22. SSL ciphers supported by the sslCipherSpecs configuration option (continued)

Cipher	Decimal value	Hexadecimal value	SSL value	Description
DES_SHA_EXPORT	512	x00000200	09	56-bit DES encryption with SHA-1 message authentication and RSA key exchange
DH_DSS_AES_128_SHA	1048576	x00100000	30	128-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_AES_256_SHA	65536	x00010000	36	256-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_DES_SHA	128	x00000080	0C	56-bit DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_TRIPLE_DES_SHA	8	x00000008	0D	168-bit 3DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_RSA_AES_128_SHA	2097152	x00200000	31	128-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_AES_256_SHA	131072	x00020000	37	256-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_DES_SHA	64	x00000040	0F	56-bit DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_TRIPLE_DES_SHA	4	x00000004	10	168-bit 3DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DHE_DSS_AES_128_SHA	4194304	x00400000	32	128-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_DSS_AES_256_SHA	262144	x00040000	38	256-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate

Table 22. SSL ciphers supported by the sslCipherSpecs configuration option (continued)

Cipher	Decimal value	Hexadecimal value	SSL value	Description
DHE_DSS_DES_SHA	32	x00000020	12	56-bit DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_DSS_TRIPLE_DES_SHA	2	x00000002	13	168-bit 3DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_RSA_AES_128_SHA	8388608	x00800000	33	128-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_AES_256_SHA	524288	x00080000	39	256-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_DES_SHA	16	x00000010	15	56-bit DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_TRIPLE_DES_SHA	1	x00000001	16	168-bit 3DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
RC2_MD5_EXPORT	4096	x00001000	06	40-bit RC2 encryption with MD5 message authentication and RSA key exchange
RC4_MD5_EXPORT	8192	x00002000	03	40-bit RC4 encryption with MD5 message authentication and RSA key exchange
RC4_MD5_US	2048	x00000800	04	128-bit RC4 encryption with MD5 message authentication and RSA key exchange
RC4_SHA_US	1024	x00000400	05	128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange
RSA_AES_128_SHA	16384	x00004000	2F	128-bit AES encryption with SHA-1 message authentication and RSA key exchange
RSA_AES_256_SHA	32768	x00008000	35	256-bit AES encryption with SHA-1 message authentication and RSA key exchange

Table 22. SSL ciphers supported by the `sslCipherSpecs` configuration option (continued)

Cipher	Decimal value	Hexadecimal value	SSL value	Description
TRIPLE_DES_SHA_US	256	x00000100	0A	168-bit 3DES encryption with SHA-1 message authentication and RSA key exchange

Setting up an LDAP Client

As with the LDAP server, the LDAP client that chooses to use SSL/TLS protected communication needs access to a key database. If the LDAP server you are going to contact is using a self-signed certificate (as is done frequently while testing SSL/TLS protected communications between an LDAP client and server), then the self-signed certificate of the LDAP server must be stored into the LDAP client's key database.

If the LDAP server you are going to contact is using a certificate which is signed by a certificate authority (CA), you must ensure that the certificate for the CA is contained in the key database. Use whatever means is provided by the CA for obtaining the CA certificate. The certificate should be obtainable in a format that is acceptable to the GSKKMAN utility.

If the LDAP server is configured for server and client authentication and the client wants client authentication to occur, then the LDAP client must obtain its own certificate from a CA and store it in the client's own key or key database and mark it as the default.

After the key database file is created and contains the proper certificates, then the LDAP client is ready to perform SSL/TLS protected communications with an LDAP server. The LDAP client utilities (for example, LDAPSrch) can be used to communicate securely with the LDAP server using a secure only connection. The utilities are explained in [z/VM: TCP/IP User's Guide](#).

Support of Certificate Bind

The SASL bind mechanism of EXTERNAL is supported by the LDAP server. This means that the authentication on the bind is performed using the data obtained during the SSL/TLS client authentication that was performed on the SSL/TLS handshake with the client.

To use SASL External bind, the following steps must occur:

- The LDAP server must be configured and started with `sslAuth` set to `serverClientAuth` so that the server can authenticate the client.
- The client connects to the LDAP server and performs the SSL/TLS handshake. The handshake sends the client certificate to the LDAP server.
- The client performs a SASL bind with the mechanism of **EXTERNAL**.

At this point, the LDAP server will consider the bind DN of the client for authorization purposes to be the client's DN as transmitted in the client's certificate on the handshake. The name specified in the BIND request must match the subject name in the client certificate or must be null.

Configuring for Encryption or Hashing

The LDAP server allows the **userPassword**, **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, **ibm-slappedAdminPw** and **ibm-slappedMasterPw** attribute values to be encrypted or hashed when stored in the directory. This prevents clear data from being accessed by users, including the system administrators. An administrator may configure the server to encrypt or hash **userPassword** or **ibm-slappedAdminPw** attribute values in either a one-way hashing format or a two-way symmetric encryption format. **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-slappedMasterPw** attribute values can only be encrypted in a two-way symmetric encryption format. Besides encryption or hashing, access to data stored in the directory can be protected by the directory access control mechanism.

One-way Hashing Formats

The one-way hashing formats include crypt, MD5, SHA, Salted SHA (SSHA), SHA-2, and Salted SHA-2. The SHA-2 and Salted SHA-2 hashing algorithms consist of the following methods: SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, and SSHA512 (Salted SHA512).

During a simple bind, the bind password is hashed using the appropriate algorithm and compared with the stored hashed **userPassword** or **ibm-slappedAdminPw** attribute value. Assuming that a client is authorized using directory access controls to see the **userPassword** or **ibm-slappedAdminPw** attribute values on a search, the values are returned to the client in one of the following manners:

1. If the **pwSearchOutput** configuration option is set to **binary** and the **userPassword** or **ibm-slappedAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the binary hash.
2. If the **pwSearchOutput** configuration option is set to **base64** and the **userPassword** or **ibm-slappedAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the base64-encoded binary hash.
3. If the **userPassword** or **ibm-slappedAdminPw** attribute value is hashed in crypt, the LDAP server returns the encryption tag ({crypt}) in UTF-8 followed by the binary hash and sent over the wire in UTF-8. The **pwSearchOutput** configuration option has no bearing on the retrieval of crypt hashed password values.
4. If the **userPassword** or **ibm-slappedAdminPw** attribute value is hashed in SHA-2 (SHA224, SHA256, SHA384, or SHA512), the LDAP server returns the encryption tag (for example, {SHA224}) in UTF-8 followed by the base64-encoded binary hash.. The **pwSearchOutput** configuration option has no bearing on the retrieval of SHA-2 hashed password values.
5. If the **userPassword** or **ibm-slappedAdminPw** attribute value is hashed in Salted SHA (SSHA) or Salted SHA-2 (SSHA224, SSHA256, SSHA384, or SSHA512), the LDAP server returns the encryption tag (for example, {SSHA}) in UTF-8 followed by the base64-encoded binary hash and salt value. The **pwSearchOutput** configuration option has no bearing on the retrieval of Salted SHA (SSHA) or Salted SHA-2 hashed password values.

The following are examples of retrieving the same SHA hashed **userPassword** attribute value, using the LDAPSrch client utility with the **-L** option specified. For additional information about the LDAPSrch client utility, see [z/VM: TCP/IP User's Guide](#).

When the **pwSearchOutput** configuration option is set to **binary**, the SHA hashed **userPassword** value is displayed by LDAPSrch as follows:

```
userpassword: : e1NIQX2pmT42RwaBaro+JXF4UMJsnNDYnQ==
```

The **userPassword** attribute value that is returned above contains both the UTF-8 encryption tag, {SHA}, and the binary hashed data, but they have been base64-encoded by the LDAPSrch client utility to present the value in a printable format.

When the **pwSearchOutput** configuration option is set to **base64**, the SHA hashed **userPassword** value is displayed by LDAPSrch as follows:

```
userpassword: {SHA}qZk+NkcGgWq6PiVxeFDCbJzQ2J0=
```

The **userPassword** attribute value that is returned above is displayed as sent to the LDAPSrch client utility because the hashed binary data after the {SHA} encryption tag has already been base64-encoded by the LDAP server.

For applications requiring retrieval of clear text passwords or data, such as middle-tier authentication agents, the directory administrator must configure the LDAP server to perform either a two-way encryption or no encryption of user passwords or data.

Two-way Encryption Formats

The supported two-way encryption formats include DES and AES. Two-way encryption allows the values of the **userPassword**, **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, or **ibm-slappedAdminPw** and **ibm-slappedMasterPw** attributes to be retrieved as part of an entry in the original clear format. Some applications, such as middle-tier authentication servers, require passwords to be retrieved in clear text while installation security policies might prohibit storing clear passwords in secondary permanent storage. This option satisfies both requirements. An encrypted password can be used for password matching on a simple bind and can be decrypted for return as clear text on a search request. During simple bind, the stored encrypted **userPassword** or **ibm-slappedAdminPw** attribute values are decrypted and compared with the bind password. During a search, if the client is authorized using directory access controls to see the **userPassword**, **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, **ibm-slappedAdminPw**, or **ibm-slappedMasterPw** attribute values, then they are decrypted and returned as clear text.

Symmetric Encryption Keys

The DES and AES encryption format use symmetric encryption keys. DES uses 56-bit (single length), 112-bit (double length) or 168-bit (triple length) keys while AES uses 128-bit, 192-bit, or 256-bit keys. A DES or AES key can be stored in a sequential file referenced by the LDAPKEYS FILEDEF statement.

DES and AES keys can be stored in a sequential file referenced by the LDAPKEYS FILEDEF statement. The file consists of fixed-length or variable-length records with a maximum record length of 255. The records are assumed to be in the IBM-1047 code page. Comment records begin with '#' or '*' and blank records are ignored. Each record in the file defines a single key and has the following format:

```
key-label key-part-1 key-part-2 key-part-3 key-part-4
```

The fields are separated by one or more blanks. Each key part consists of 16 hexadecimal characters representing 8 bytes of the key. A DES key requires the key label and the one, two or three key parts while an AES key requires the key label and all four key parts. In a DES key, the low-order bit in each byte is a parity bit. The parity bit must be set so that there is an odd number of 1s in each byte, but the bit is not used for encryption. Therefore, DES uses 56-bits out of each 8-byte key part for encryption. An AES key does not use parity bits, so the entire key (256 bits) is used for encryption.

Configuring for user and administrator password encryption or hashing

The LDAP server allows prevention of unauthorized access to user or administrator passwords in the LDBM and CDBM backends. The **userPassword** and **ibm-slappedAdminPw** attribute values can be encrypted or hashed when stored in the directory, which prevents clear text passwords from being accessed by any users, including the system administrators. Use of the terms "user password" and "password" pertain to the **userPassword** attribute. Use of the term "user entry" refers to an entry in LDBM or CDBM that contains a **userPassword** attribute. Use of the terms "administrator password" and "password" pertain to the **ibm-slappedAdminPw** attribute. Use of the term "administrator entry" refers to an entry in LDBM or CDBM that contains an **ibm-slappedAdminPw** attribute.

An administrator may configure the server to encrypt or hash **userPassword** and **ibm-slappedAdminPw** attribute values in either a one-way hashing format or a two-way, symmetric, encryption format. The **pwEncryption** configuration option in an LDBM or CDBM section of the LDAP server configuration file specifies the encryption method that is to be used to encrypt or hash the **userPassword** and **ibm-slappedAdminPw** attribute values in that LDBM or CDBM backend. For more information about the password encryption or hashing types, see the [pwEncryption](#) option.

After the server is configured and started, any user passwords or administrator passwords for new user or administrator entries or modified passwords for existing user or administrator entries are encrypted or hashed before they are stored in either the LDBM or CDBM backend. The encrypted or hashed passwords are tagged with the encryption algorithm name so that passwords encrypted or hashed in different formats can coexist in the directory. If a tagged encrypted or hashed **userPassword** or **ibm-slappedAdminPw** attribute value is present on an add or modify operation, the attribute value is

added as it is, with no additional encryption or hashing performed on the value even if the **pwEncryption** configuration option is set to a different type of encryption or hashing.

If the **pwEncryption** configuration option in an LDBM or CDBM backend is changed, existing passwords remain unchanged and continue to be usable. In other words, existing user and administrator password values are not automatically converted to the new encryption method or key label.

The **db2pwden** utility is provided as a migration utility to encrypt or hash all unencrypted, AES encrypted, or DES encrypted **userPassword** attribute values in the encryption or hashing method specified by the **pwEncryption** configuration option in the LDBM or CDBM backend. The **db2pwden** utility does not convert encrypted or hashed **ibm-slappedAdminPw** attribute values. For example, the **db2pwden** utility allows an LDAP administrator to convert passwords from AES to DES, DES to AES, or AES to crypt. The **db2pwden** utility is similar to the LDAP client utilities, such as **ldapsearch**, in that it acts as a client to the LDAP server and has similar command-line parameters. For more information about the **db2pwden** utility, see “DB2PWDEN (db2pwden utility)” on page 172. For information about LDAP client utilities, such as **ldapsearch**, see *z/VM: TCP/IP User's Guide*. The **db2pwden** utility must be run by an LDAP administrator with the appropriate authority or a user with the authority to update **userPassword** values. See Administrative group and roles in *z/VM: TCP/IP LDAP Administration Guide* for more information about administrative role authority.

If the **pwEncryption** configuration option is changed from AES or DES encryption to another encryption or hashing method or to a different AES or DES key label, the LDAP server must have access to the original AES or DES key label so that decryption of existing **userPassword** and **ibm-slappedAdminPw** values still occurs on bind, search, and compare operations. If you want to remove the LDAP server's access to the original AES or DES key label, it is necessary to migrate all existing AES or DES **userPassword** values to the new encryption or hashing method or new AES or DES key label by using the **db2pwden** utility. To migrate all existing AES or DES **ibm-slappedAdminPw** values to the new encryption method or new AES or DES key label, each entry that contains an **ibm-slappedAdminPw** value must be searched to obtain its value. Then, these entries must be manually modified using the **ldapmodify** utility to replace the existing value with the same value so that the new encryption or hashing method is used. After all AES or DES encrypted passwords are converted to the new encryption or hashing method or new AES or DES key label, the LDAP server's access to the original AES or DES key label can be removed.

A simple bind succeeds if the password provided in the bind request matches with any of the multiple values of the **userPassword** attribute. A simple bind succeeds with an administrator entry if the password provided in the bind request matches the single **ibm-slappedAdminPw** attribute value. Note that depending on when **userPassword** or **ibm-slappedAdminPw** values are stored in the directory, different attribute values can be encrypted or hashed using different encoding methods.

For information about the unloading of **userPassword** and **ibm-slappedAdminPw** attribute values in the **ds2ldif** utility, see “DS2LDIF (ds2ldif utility)” on page 175.

Note:

1. The LDAP server does not permit **userPassword** or **ibm-slappedAdminPw** attributes in distinguished names.
2. Some important considerations for password encryption or hashing and basic replication are described in Password encryption and replication in *z/VM: TCP/IP LDAP Administration Guide*. If **userPassword** or **ibm-slappedAdminPw** attribute values are replicated in an advanced replication environment, the attribute values are replicated in the clear no matter the **pwEncryption** configuration option setting. Use a secure SSL connection between the supplier and consumer servers to protect this sensitive data.
3. The **crypt()** algorithm, implemented across many platforms, accepts only the first eight characters of a password. As a result, any password supplied on an **ldap_simple_bind()** or **ldap_compare()** API that matches the first eight characters of a **userPassword** or **ibm-slappedAdminPw** attribute value hashed with the crypt algorithm in the directory matches.

Configuring for Secret Encryption

The LDAP server allows prevention of unauthorized access to **secretKey** and **replicaCredentials** in the LDBM and CDBM backends. The **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-**

slapdMasterPw attribute values can be encrypted when stored in the directory, which prevents clear text passwords and data from being accessed by any users, including the system administrators. Use of the term *secret encryption* refers to any entry in LDBM or CDBM that contains **secretKey**, **replicaCredentials**, **ibm-replicaKeyPwd**, or **ibm-slapdMasterPw** attributes.

The administrator may configure an LDBM or CDBM backend in the server to encrypt the **secretKey** and **replicaCredentials** attribute values in DES or AES by specifying the **secretEncryption** option in the LDBM or CDBM backend section of the LDAP server configuration file. For more information on encrypting **secretKey** and **replicaCredentials** attribute values, refer to [secretEncryption](#).

After the server is configured and started, any **secretKey** and **replicaCredentials** attribute values for new entries or modified **secretKey** and **replicaCredentials** attribute values for existing entries are encrypted before they are stored in either the LDBM or CDBM backend. The encrypted **secretKey** and **replicaCredentials** attribute values are tagged with the encoding algorithm name so that values encrypted in different formats can coexist in the directory. When the **secretEncryption** option is changed, existing attribute values remain unchanged and continue to be usable.

If the **secretEncryption** configuration option in an LDBM or CDBM backend is changed, existing secret encryption attribute values remain unchanged and continue to be usable. In other words, existing secret encryption attribute values are not automatically converted to the new encryption method or key label.

If the **secretEncryption** configuration option is changed from AES to DES, DES to AES, or to a different AES or DES key label, the LDAP server must have access to the original AES or DES key label so that decryption of existing values still occurs on bind, search, and compare operations. If you want to remove the LDAP server's access to the original AES or DES key label, it is necessary to migrate all existing AES or DES encrypted secret encryption values to the new AES or DES key label. This is accomplished by using LDAPSrch to retrieve all the entries that have secret encryption values. For each entry that is returned by LDAPSrch, use LDAPMDfy to change the secret encryption to the same value returned on LDAPSrch. By modifying secret encryption values in this manner, the LDAP server re-encrypts the values using the new AES or DES key label that is specified on the **secretEncryption** configuration option. After the conversion of all secret encryption values are completed, the LDAP server's access to the original AES or DES key label can be removed.

For information on the unloading of **secretKey** and **replicaCredentials** attribute values, see [“DS2LDIF \(ds2ldif utility\)”](#) on page 175.

Note: Some important considerations for secret encryption or hashing and basic replication are described in Data encryption or hashing and basic replication. If secret encryption attribute values are replicated in an advanced replication environment, the attribute values are replicated in the clear no matter the **secretEncryption** configuration option setting. Use a secure SSL connection between the supplier and consumer servers to protect this sensitive data. See *z/VM: TCP/IP LDAP Administration Guide*. for more information on password encryption and basic replication.

Configuring Plug-in Extensions

To configure plug-ins, specify the **plugin** configuration file option in DS CONF. See [plugin](#).

For information about creating and setting up a plug-in extension, see *IBM Tivoli Directory Server Plug-in Reference for z/OS*.

Example Configuration Scenarios

This section shows scenarios of LDAP server configurations. Only some of the options that can be specified for each section of the LDAP server configuration file are shown. See [Table 26 on page 130](#) for a complete list of the options that are available for each section.

Configuring SDBM and LDBM Backends

The configuration example in this section uses the SDBM and LDBM backends to show the configuration file checklist next to the corresponding sample configuration file.

Table 23. Sample checklist and DS CONF (using SDBM and LDBM)

Section	Check	Sample DS CONF
Global	✓	<pre># Filename DS CONF # Global section sizelimit 500 timelimit 3600 adminDn "cn=LDAP Administrator,o=My Company" listen ldap://:389</pre>
SSL/TLS		
SDBM backend	✓	<pre># SDBM backend section database sdbm GLDBSD31 suffix "cn=racfu01,o=ibm,c=us"</pre>
LDBM backend	✓	<pre># LDBM backend section database ldbm GLDBLD31 suffix "o=My Company"</pre>
Password encryption		
Replication		
Native authentication		

Configuring LDBM with Native Authentication and GDBM Backends

The configuration example in this section uses the LDBM and GDBM backends to show the configuration file checklist next to the corresponding sample configuration file. The GDBM backend is based on the Byte File System.

Table 24. Sample checklist and DS CONF (using GDBM and LDBM)

Section	Check	Sample DS CONF
Global	✓	<pre># Filename DS CONF # Global section sizelimit 500 timelimit 3600 adminDn "cn=LDAP Administrator,o=My Company" listen ldap://:389</pre>
SSL/TLS		
GDBM backend	✓	<pre># GDBM backend section database gdbm GLDBGD31</pre>
Password encryption		
Replication		
Native authentication		
LDBM backend	✓	<pre># LDBM backend section database ldbm GLDBLD31 suffix "o=My Company" usenativauth all</pre>
Password encryption		

Table 24. Sample checklist and DS CONF (using GDBM and LDBM) (continued)

Section	Check	Sample DS CONF
Replication		
Native authentication	✓	nativeauthsubtree all

Configuring RACF/VM Change Logging with SDBM and GDBM Backends

The configuration example in this section uses the SDBM and GDBM backends to show the configuration file checklist next to the corresponding sample configuration file.

Table 25. Sample checklist and DS CONF (using SDBM and GDBM)

Section	Check	Sample DS CONF
Global	✓	<pre># Filename DS CONF # Global section sizelimit 500 timelimit 3600 adminDn "cn=LDAP Administrator,o=My Company" listen ldap://:389 listen ldap://:pc</pre>
SSL/TLS		
SDBM backend	✓	<pre># SDBM backend section database sdbm GLDBSD31 suffix "cn=racfu01,o=ibm,c=us"</pre>
GDBM backend	✓	<pre># GDBM backend section database gdbm GLDBGD31 changeLogging on changeLoggingParticipant on databaseDirectory /var/ldap/gdbm changeLogMaxAge 86400 changeLogMaxEntries 1000</pre>
Password encryption		
Replication		
Native authentication		

Note: To perform the RACF configuration required to support creation of LDAP change log entries, see [How to set up and use the LDAP Server for logging changes in the z/VM: TCP/IP LDAP Administration Guide](#) and [Activating LDAP Change Notification in the z/VM: RACF Security Server Security Administrator's Guide](#).

Configuration File (DS CONF) Format and Configuration Options

The DS CONF file consists of the following sections:

Global section

Contains configuration options that apply to the LDAP server as a whole (including all backends).

SDBM backend-specific section

Contains configuration options that apply to the SDBM backend.

LDBM backend-specific section

Contains configuration options that apply to the LDBM backend. It is possible to have one or more of these sections depending on how many LDBM backends your installation uses.

GDBM backend-specific section

Contains configuration options that apply to the GDBM change log backend.

CDBM backend-specific section

Contains configuration options that apply to the file-based CDBM backend.

Figure 2 on page 128 shows the general format of DS CONF.

```
# Global options - these options apply to every database
<global configuration options>
#
# SDBM database definition and configuration options
database SDBM GLDBSD31
<configuration options specific to SDBM backend>
#
# LDBM database definition and configuration options
database LDBM GLDBLD31
<configuration options specific to LDBM backend>
#
# File-based GDBM database definition and configuration options
database GDBM GLDBGD31
<configuration options specific to file-based GDBM backend>
#
# CDBM database definition and configuration options
database CDBM GLDBCD31
<configuration options specific to CDBM backend>
```

Figure 2. General format of DS CONF

Noted below are some rules for setting up DS CONF:

- The configuration file contains a global section containing options that apply to the entire LDAP server, followed by one or more database backend sections that contain options that apply to a specific backend. Each backend section begins with a **database** option and continues to the next **database** option. The global section starts at the beginning of the configuration file and ends at the first **database** option. The **sizelimit** and **timelimit** options can be either global or specific to a backend: they are global if they appear in the global section and are specific to a backend if they appear in a backend section. See the descriptions of these options for more information.
- The configuration file must be in code page IBM-1047.
- The maximum length of a line in the configuration file is 1024 characters.
- Each configuration line consists of an option and a value separated by one or more blanks. Begin each configuration option in column one. The option is not case-sensitive. The value might or might not be case-sensitive depending upon the option. If an argument contains one or more blank spaces, the value should be enclosed in double quotation marks (for example, "value one"). If a value contains a double quotation mark or a backslash character (\), the double quotation mark or backslash character should be preceded by a backslash character (\).
- A line that begins with a space or tab character in column one is considered a continuation of the previous line. Everything after the space or tab character is appended to the previous line. The maximum length of the initial line plus all continuation lines is 1024 characters.
- A line that begins with a pound sign (#) in column one is a comment line. Comment lines can be continued and are ignored.
- A pound sign (#) can be used to add a comment to the end of a configuration line. The pound sign (#) must be separated from the option value by at least one blank. Anything following the pound sign (#) will be ignored, including any continuation lines. A pound sign (#) will not be treated as the start of a comment if it occurs within a quoted value.
- Blank lines can be used to separate configuration lines.
- Options that expect a value of **on** or **off** will also accept **yes**, **y**, **no** and **n**. The option value is not case-sensitive.
- For single-valued options that appear more than once, the last appearance in the DS CONF file is used.

Specifying a Value for Filename

For configuration file options that contain the *filename* value, the value can be specified in one of the following ways:

/pathname/filename

Specifies the full path name of a file in the Byte File System.

filename

Specifies a file name in the Byte File System that is relative to the current working directory of the LDAP server. The LDAP server sets the current working directory to the value of the **HOME** environment variable or to `/etc/ldap` if the **HOME** environment variable is not defined. This format is not recommended.

//filename.filetype.filemode

Specifies the name of a CMS file. *filemode* is optional; the default is `A`.

//DD:DDNAME

Specifies the DDNAME for a FILEDEF or OPENVM PATHDEF command that defines the file. The file can be a BFS or CMS file.

Specifying a Value for a Distinguished Name

The value for the following configuration options is a distinguished name (DN): **adminDN**, **masterServerDN**, **peerServerDN**, **nativeAuthSubtree**, and **suffix**. Special characters (as identified in RFC 2253) used in the DN must be properly escaped using two back slashes (`\\`). Note that the double back slashes are only needed in the configuration file; in all other usages, the special characters are usually prefixed by a single backslash. For exceptions to this when using SDBM, see [Accessing RACF information in z/VM: TCP/IP LDAP Administration Guide](#). For valid DN formats, see IETF RFC 2253, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*.

For example, to use a RACF user ID `#1admin` as the LDAP root administrator defined in the configuration file, the **adminDN** configuration option would look similar to:

```
adminDN "racfid=\\#1admin,profiletype=user,cn=myRacf"
```

With LDAP 3 support, UTF-8 characters can be used for textual attributes stored in the directory. It is also desirable to allow any UTF-8 character to appear in distinguished names, and in particular, the **adminDN** distinguished name. Because the LDAP configuration files are defined to hold information only in the IBM-1047 character set, a solution is required for the configuration files that allows you to use distinguished names containing UTF-8 characters but using only the IBM-1047 character set. To solve this problem, an escape mechanism has been introduced for purposes of entering a DN. This escape mechanism allows the entry of UTF-8 characters while keeping the input string value to within the IBM-1047 character set. The escape mechanism employed requires that you express UTF-8 characters which are not within the `X'00' - X'7F'` range (7-bit ASCII which is the single-byte form of UTF-8 characters) in the form of a set of four character representations. This representation has the form `"&nmm"` where $0 < n < 3$ and $0 < m < 7$. You might recognize *nmm* as being an octal value for a byte of information. Thus, if you want to create the following distinguished name:

```
cn=Peter <U umlaut>nger, o=Widgets, c=DE
```

specify the DN as:

```
cn=Peter &nmm&nmmnger, o=Widgets, c=DE
```

Because the `<U umlaut>` is not within the 7-bit ASCII range, the value must be escaped to the octal representation of the UTF-8 multi-byte character. In the case of `<U umlaut>`, the Unicode code point is `X'00DC'`. Converted to UTF-8, this character is a multi-byte sequence: `X'C3BC'`. (Refer to [“UTF-8 Support” on page 190](#) for conversion information.) Converted to the escaped form for input into the DN field, this

character is represented as "&303&234" since X'C3' is octal 303 and X'BC' is octal 234. Therefore, the DN above would be entered as:

```
cn=Peter &303&234nger, o=Widgets, c=DE
```

If there is a case where you need to enter a DN which contains the string "&nmm" where $0 < n < 3$ and $0 < m < 7$, then you must escape the ampersand by using its octal representation which is "&046".

The following are restrictions concerning the attributes used in a DN:

Note:

1. The default matching rule for an attribute used in a DN that is not contained in the LDAP server schema is **caseIgnoreMatch**. This results in this part of the DN being normalized by using uppercase in the attribute value. If the attribute is later defined in the schema and the matching rule is not **caseIgnoreMatch**, that part of the DN may now be normalized differently. Therefore, the normalized version of the DN used in an operation may not match the normalized version of the DN in the configuration file, resulting in the failure of the operation.
2. Do not use an attribute with Integer syntax (1.3.6.1.4.1.1466.115.121.1.27) in a DN in the configuration file, especially in a suffix. Integer attribute values in a normalized DN have a special format that might cause problems within the LDAP server.

In particular, if an attribute used in a suffix is not in the LDAP server schema and is later added to the schema with a different matching rule, then restart the LDAP server after the schema definition is added. Otherwise, operations using that suffix can fail.

Configuration File Checklist

The following table is provided to assist you in determining which configuration file options you will need to use in your DS CONF file. Depending on the section in the configuration file (Global, SDBM, GDBM, CDBM, or LDBM), certain topics (SSL, schema, replication, and so on) have options that are required or optional.

Table 26. Configuration file options checklist

Section/topic	Check	Options
Global		adminDN is required. adminPW, allowAnonymousBinds, altServer, audit, blockedConnectionTimeout, commThreads, digestRealm, dnCacheSize, idleConnectionTimeout, include, listen, logfile, maxConnections, operationsMonitor, operationsMonitorSize, pcIdleConnectionTimeout, pcThreads, plugin, pwSearchOutput, referral, schemaPath, schemaReplaceByValue, securityLabel, sendV3stringsoverV2as, serverEtherAddr, sizeLimit, srvStartUpError, tcpTerminate, timeLimit, and validateincomingV2strings are optional.
SSL/TLS		sslKeyRingFile is required if a listen option is initialized for secure socket communications or a listen option is initialized for non-secure socket communications that is intended to support switching to secure socket communications once the connection is established. sslAuth, sslCertificate, sslCipherSpecs, sslKeyRingFilePW, and sslKeyRingPWStashFile are optional.

Table 26. Configuration file options checklist (continued)

Section/topic	Check	Options
Activity logging		logfile is required. logfileFilter , logfileMicroseconds , logfileMsgs , logfileOps , logfileRecordType , logfileRolloverDirectory , logfileRolloverSize , logfileRolloverTOD , and logfileVersion are optional.
SDBM backend		database and suffix are required. enableResources, include, readOnly, sizeLimit, and timeLimit are optional.
Attribute encryption or hashing		pwCryptCompat, pwEncryption, and secretEncryption are optional.
Basic replication		Either peerServerDN or both masterServer and masterServerDN are required. peerServerPW and masterServerPW are optional.
Native authentication		useNativeAuth and nativeAuthSubtree are required. nativeUpdateAllowed is optional.
LDBM backend		database and suffix are required. attrOverflowCount , changeLoggingParticipant , commitCheckpointEntries , commitCheckpointTOD , databaseDirectory , extendedGroupSearching , fileTerminate , filterCacheBypassLimit , filterCacheSize , include , persistentSearch , readOnly , sizeLimit , and timeLimit are optional.
GDBM backend (file-based)		database is required. changeLogging , changeLoggingParticipant , changeLogMaxAge , changeLogMaxEntries , commitCheckpointEntries , commitCheckpointTOD , databaseDirectory , fileTerminate , filterCacheBypassLimit , filterCacheSize , include , persistentSearch , readOnly , sizeLimit , and timeLimit are optional.
CDBM backend		database is required. attrOverflowCount , changeLoggingParticipant , commitCheckpointEntries , commitCheckpointTOD , databaseDirectory , extendedGroupSearching , fileTerminate , filterCacheBypassLimit , filterCacheSize , include , persistentSearch , readOnly , sizeLimit , and timeLimit are optional.

Note: The **adminDN** configuration option is required and specifies the LDAP root administrator in the configuration file. The password for this LDAP root administrator can be specified in a directory entry or in the **adminPW** configuration option. See “Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85 for more information. Note the use of the **adminPW** configuration option is discouraged. Instead, an existing entry in the directory is designated as the **adminDN**.

If the distinguished name specified for the **adminDN** configuration option does not reside within a configured backend suffix and the password is specified in the **adminPW** configuration option, password policy is not supported for the LDAP root administrator defined in the configuration file.

Configuration File Options

This section contains an alphabetical listing of the configuration file options. For each option, a table shows an **X** in the areas (Global, LDBM, SDBM, GDBM, and CDBM) of the configuration file where the option can be used.

aclSourceCacheSize *num-entries*

Global	LDBM	SDBM	GDBM	CDBM
			X	

Specifies the maximum number of entries to store in the ACL Source cache. This cache holds information regarding ACL definitions within the database. Retrieval of information from this cache avoids database read operations when resolving access permissions.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.

The default is 100.

adminDN *dn*

Global	LDBM	SDBM	GDBM	CDBM
X				

The distinguished name (DN) of the root administrator for this LDAP server. Typically, this DN has unrestricted access to all entries in the directory except for entries in backends that are read-only replicas. When the LDAP server is in maintenance mode, the LDAP root administrator has unrestricted access to all entries in the directory. Select a name that is descriptive of the person that knows and administers the LDAP server. The format of the name must be in DN format that is described in *Data Model in z/VM: TCP/IP LDAP Administration Guide*. you might want the DN to have the same suffix as one of the suffix option values in the configuration file.

[“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords”](#) on page 85 describes how to set up this root administrator DN. Additional root administrators can be defined using the administrative group and assigning the root administrator role. See [Administrative group and roles](#) in *z/VM: TCP/IP LDAP Administration Guide* for more information.

For information on specifying a value for a distinguished name for this option, see [“Specifying a Value for a Distinguished Name”](#) on page 129.

adminPW *string*

Global	LDBM	SDBM	GDBM	CDBM
X				

The password of the root administrator (adminDN) for this server.

[“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords”](#) on page 85 describes how to set up your administrator password.

Note: Use of the adminPW configuration option is strongly discouraged in production environments. Instead, specify your adminDN as the distinguished name of an existing entry in the directory information tree. This will eliminate passwords from the configuration file.

allowAnonymousBinds {on | off}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies whether an LDAP client can perform unauthenticated operations on the LDAP server. If **off**, clients must explicitly bind to the server with a distinguished name. If **on**, a client may access the server without binding with a distinguished name and will have access to data as a member of the cn=anybody group. For more information on access control of directory data, see [Using Access Control](#) in *z/VM: TCP/IP LDAP Administration Guide*.

The default is **on**.

altServer ldap_URL

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies an equivalent server to this LDAP server. It may or may not be a replica, but should contain the same naming contexts. There is no required format for the value, however, LDAP URL format is most commonly used and supported by LDAP clients. See [listen](#) for a description of LDAP URL format. The option may be specified multiple times to define more than one alternate server. The alternate servers are placed in the **altServer** attribute in the rootDSE and can be queried by LDAP clients to determine other servers that may be contacted in case this server is not available at some later time.

In the following example, myldap.server.com is the host name and 3389 is the port number of the LDAP directory URL:

```
altServer ldap://myldap.server.com:3389
```

In the following example,

```
5f1b:df00:ce3e:e200:20:800:2078:e3e3
```

is the IPv6 address and 389 is the port number of the LDAP URL:

```
altServer ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

attrOverflowCount count

Global	LDBM	SDBM	GDBM	CDBM
	X			X

For LDBM and CDBM, specifies the number of attribute values required to store the attribute values in an internal indexed table, providing quicker access to the values of large multi-valued attributes such as group membership lists.

The value must be either 0 or in the range 64 to 2147483647. A value of 0 disables attribute overflow based on the attribute value count.

The default is 512.

audit {on | off | all,operations | error,operations | none,operations}

Global	LDBM	SDBM	GDBM	CDBM
X				

Turns LDAP auditing on or off and specifies which operations are to be audited and the associated audit level. When auditing is on, an LDAP SMF type 83 subtype 3 audit record is generated for an operation if the operation is specified on an **audit** option and the operation result matches the audit level.

This option can be specified multiple times, once to turn auditing on or off and one or more times for each audit level to specify the operations to audit for that level. Multiple operations can be specified for a level by either putting a **+** between them on the **audit** option or by specifying multiple **audit** options with the same level.

Operations can be audited only when they fail or all the time. The following audit levels are supported:

all

An LDAP audit record will be generated for the specified operations.

error

An LDAP audit record will be generated for the specified operations when they fail.

none

An LDAP audit record is not generated for the specified operations.

The supported values for *operations* can be one or more of: **add, bind, compare, connect, delete, disconnect, modify, modifydn, search, unbind**.

If an operation is specified in more than one level, the last level is used for the operation. If an operation is not specified in any level, the level defaults to **none** for that operation.

The LDAP server AUDIT operator command can be used to change the audit settings and to turn audit on or off while the LDAP server is running. See [“Dynamic Server Operation” on page 161](#) for more information.

The default is **off**.

For example, the following **audit** options turn auditing on for modify, search, and bind failures and for all add operations. The other operations are not audited.

```
audit error,modify+search+bind
audit all,add
audit on
```

blockedConnectionTimeout *num-seconds*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the amount of time in seconds that the LDAP server will wait for a blocked send operation to complete on a client connection. When a send operation times out, the client connection is dropped. This option is useful to avoid server commThreads from being consumed by client applications that fail and stop receiving responses. If too many, or all threads get blocked indefinitely, a denial of service condition occurs.

The blocked connection is detected when an EWOULDBLOCK condition is returned (or GSK_WOULDBLOCK condition for SSL connections). This may not occur until some time after an errant client stops receiving data and the TCP buffers fill. Therefore, it is possible for a blocked connection to remain active slightly longer than the blockedConnectionTimeout value.

The value must be between 0 and 2678400 (equal to 31 days). A value of 0 indicates that a blocked connection remains active indefinitely.

The default is **0** (indefinitely).

changeLogging {on | off}

Global	LDBM	SDBM	GDBM	CDBM
			X	

Turns change logging on or off.

When change logging is on, all change logging operations are allowed. When change logging is off, change log entries can be searched, modified, and deleted, but no new change log entries can be created and no automatic trimming of the change log is performed.

The default is **on**.

changeLoggingParticipant {on | off}

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Allows or disallows change logging for changes made to entries in this backend. When specified in GDBM, **changeLoggingParticipant** controls the logging of modifications to the LDAP server schema entry.

Note: This option does not turn on or off change logging. That is done by the **changeLogging** option.

The default is **on**.

changeLogMaxAge *nnn*

Global	LDBM	SDBM	GDBM	CDBM
			X	

Specifies the maximum age in seconds of an entry in the change log. Change log entries are deleted when they have been in the change log longer than this value, except if **changeLogging off** is specified. The value must be 60 - 2147483647. A value of 0 indicates that there is no maximum.

The default is 60.

changeLogMaxEntries *nnn*

Global	LDBM	SDBM	GDBM	CDBM
			X	X

Specifies the maximum number of entries that the change log can contain. If the number of change log entries exceeds this value and **changeLogging off** is not specified, change log entries with the lowest change numbers are deleted. Change log entries are deleted until the number of remaining entries is the maximum. The value must be 0 - 2147483647. A value of 0 indicates that there is no maximum.

The default is 0.

commitCheckpointEntries *nnn*

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies the maximum number of entries in the checkpoint file. An entry is added to the LDBM, GDBM, or CDBM checkpoint file each time a directory entry is added, changed, deleted, or renamed. When the maximum number is reached, the entries in the checkpoint file are merged into the database file and the entries are removed from the checkpoint file. The value must be 0 - 2147483647. A value of 0 indicates there is no maximum.

The default is 10,000.

commitCheckpointTOD *hh:mm*

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies a time of day at which the checkpoint file will be merged into the database file. An entry is added to the LDBM, GDBM, or CDBM checkpoint file each time a directory entry is added, changed, deleted, or renamed. Every day at the specified time, the entries in the checkpoint file are merged into the database file and the entries are removed from the checkpoint file. The value must be 00:00 - 23:59. Specify a value outside this range to disable time of day checkpoint processing.

The default is 00:00.

commThreads *num-threads*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the number of threads to be initialized for the communication thread pool. This thread pool handles the connections between the LDAP server and its clients. It is recommended that **commThreads** be set to approximately two times the number of CPUs that are running in your LPAR. However, this is a general rule depending upon the activity that your LDAP server experiences.

The default is 10.

The **commThreads** option deprecates the **maxThreads** and **waitingThreads** options, which are no longer evaluated by the LDAP server.

database *dbtype dblibpath [name]*

Global	LDBM	SDBM	GDBM	CDBM
	X	X	X	X

Marks the beginning of a new database section. All global options must appear before the first database section. All options after the **database** option pertain to this backend until another **database** option is encountered.

- For *dbtype*:
 - Specify **ldbm** (file-based), **sdbm** (RACF-based), **gdbm** (file-based), or **cdbm** (file-based).
- For *dblibpath*:
 - This is the file name of the shared library (DLL) containing the backend database code. Unless you have changed the names of the LDAP DLLs, specify **GLDBLD31** when *dbtype* is **ldbm**, **GLDBSD31** when *dbtype* is **sdbm**, **GLDBGD31** when *dbtype* is **gdbm**, and **GLDBCD31** when *dbtype* is **cdbm**.
- For *name*:
 - This value is a name that is used to identify this backend. You cannot specify **schema**, **rootDSE**, or **Monitor** as the name. A name will be generated if no name is specified for a backend.

databaseDirectory name

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies the name of the directory containing the data files used by the backend to store directory data. A fully-qualified directory path must be specified. A unique directory must be specified for each backend.

The default is `/var/ldap/ldb` for an LDBM backend and `/var/ldap/gdbm` for a GDBM backend. The CDBM default is `/var/ldap/schema` if **schemaPath** is not specified; else, the **schemaPath** option setting.

digestRealm hostname

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies a realm name to be used when doing DIGEST-MD5 or CRAM-MD5 SASL authentication binds to the LDAP server. The **digestRealm** is used to help calculate a encrypt for DIGEST-MD5 and CRAM-MD5 authentication binds. It is suggested that the *hostname* be a DNS-host name and not an IP address.

If a DNS (domain name server) is accessible by the system, the default is a fully-qualified hostname of the LDAP server. Otherwise, the default is the name of the host processor.

dnCacheSize num-entries

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the maximum number of entries to store in the Distinguished Name normalization cache. This cache holds information related to the mapping of Distinguished Names between their raw form and their canonical form. Retrieval of information from this cache reduces processing required to locate entries in the database.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.

The default is 1000.

enableResources {on | off}

Global	LDBM	SDBM	GDBM	CDBM
		X		

Specifies whether the SDBM backend supports operations on RACF resources and classes. If on, SDBM accepts operations for the setropts, class, and resource profile entries. LDAP also accepts requests for creating a change log entry for a change to a RACF resource profile. If off, an SDBM search from the suffix does not return these entries and operations (including a change log request) involving these entries are rejected.

The default is **off**.

extendedGroupSearching {on | off}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

Specifies whether a backend participates in extended group membership searching on a client bind request. If this option is on, group memberships are gathered from this backend during LDAP directory bind processing in addition to the backend in which the bind DN resides. If this option is off, group memberships are not gathered from this backend unless the bind DN resides in this backend.

The group memberships gathered on a client's LDAP bind request are used for authorization checking of the client's request. The administrator should know, in general, which backends may contain group information so they can be marked for extendedGroupSearching. Group memberships are necessary for complete authorization checking of a client request. The setting of the extendedGroupSearching configuration option allows the backend to search for static and nested groups which the bind DN may belong to if the bind DN does not exist as an actual entry in the LDBM backends on the LDAP server. This applies to SASL EXTERNAL binds.

The server control authenticateOnly is supported by the LDAP server so that a client can override both extendedGroupSearching and group membership gathering from the backend where the DN resides. For more information, see [Supported Server Controls](#) in *z/VM: TCP/IP LDAP Administration Guide*.

This option applies only to the backend in which it is defined.

The default is **off**.

fileTerminate {terminate | recover}

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies whether the LDAP server ends when file system errors occur. If **terminate**, the LDAP server ends when a file system error is detected. If **recover**, the LDAP server continues processing, but the backend experiencing the file system error is set to read-only mode. No updates can be made to the directory controlled by this backend. When the problem is corrected, the backend can be reset to read-write mode using the LDAP server BACKEND operator command.

The default is **recover**.

filterCacheBypassLimit num-entries

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies the maximum number of returned entries allowed in the result set of any individual search that will be stored in the Search Filter cache. Search filters that match more than this number of entries will not be added to the Search Filter cache. This option is useful for maintaining the effectiveness of the Search Filter cache and Entry cache. It can be used to prevent a few search requests with large result sets from dominating the contents of the Entry cache.

The value must be in the range of 1 to 250. This option is ignored when the filter cache is not in use.

The default is 100.

filterCacheSize num-filters

Global	LDBM	SDBM	GDBM	CDBM
	X		X	X

Specifies the maximum number of filters to store in the Search Filter cache. This cache holds information related to the mapping of search request inputs and the result set. Retrieval of information from this cache avoids database read operations when processing search requests.

Individual search requests which return more entries than specified in the **filterCacheBypassLimit** option are not placed in the cache.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.

The LDBM default is 5000.

The GDBM default is 0.

The CDBM default is 5000.

idleConnectionTimeout *num-seconds*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the amount of time in seconds that the LDAP server waits for an idle connection or an idle paged search result set. When an idle connection times out, the client connection is dropped. When an idle paged search result set times out, the paged search result set is abandoned. Idle connections and idle paged search result sets are detected by the LDAP server's network monitor task, which checks for them every 30 seconds. Therefore, it is possible for an idle connection or idle paged search result set to remain active slightly longer than the **idleConnectionTimeout** value.

The value must be either 0 or between 30 and 2147483647. A value of 0 indicates that an idle connection or idle paged search results remains active indefinitely.

The default is 0 (indefinitely).

The recommended value is 1800 (30 minutes).

include *filename* [*serverName*]

Global	LDBM	SDBM	GDBM	CDBM
X	X	X	X	X

Specifies the path and file name of a file to be included as a part of the LDAP server configuration.

See [“Specifying a Value for Filename”](#) on page 129 for information on specifying *filename*.

Note that the LDAP server does not detect loop conditions in a set of included files. Configuration may encounter errors or fail if the same file is processed more than once. While nested include files are supported, including the same file in such a way as to form a loop condition is not supported.

serverName is the user ID of the virtual machine in which the server runs. If the optional *serverName* is specified, the include file is processed only on the specified server. This allows the LDAP server configuration files to be shared by multiple servers where each server runs in a different virtual machine. System-specific configuration information would then be placed in an include file that is processed only on the server to which it applies.

listen *ldap_URL*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies, in LDAP URL format, the IP address (or host name) and the port number where the LDAP server will listen to incoming client requests. This option may be specified more than once in the configuration file.

Note that the **listen** value may be established in the configuration file, or it may be established using the **-l** command-line parameter when starting the LDAP server (see [“LDAPSRV Command Syntax”](#) on page 80).

By default, the server listens on all available and active IPv4 addresses using port 389. The default is equivalent to `ldap://:389`.

The format of *ldap_URL* for the **listen** option to listen on a TCP/IP socket interface is the following. This format is also used for other configuration options whose value is in LDAP URL format, such as **altServer**, **masterServer**, and **referral**.

```
{ldap:// | ldaps://}[IP_address | hostname | INADDR_ANY | in6addr_any ][:portNumber]
```

The format of *ldap_URL* for the **listen** option to listen on the Program Call interface is the following:

```
ldap://:pc
```

where:

ldap://

Specifies that the server listen on nonsecure addresses or ports. Note that if SSL/TLS is configured for the server, then once a connection is established, the client may switch to secure communication using the **Start TLS** extended operation.

ldaps://

Specifies that the server listen on secure addresses or ports. Once a connection is established to the server, the client must begin the SSL/TLS handshake protocol.

IP_address

Specifies either the IPv4 or IPv6 address.

hostname

Specifies the host name. If the host name is used for the **listen** option, all of the IPv4 or IPv6 addresses associated with the *hostname* are obtained from the DNS (Domain Name Server) and the LDAP server listens on each of these IP addresses.

INADDR_ANY

Specifies the INADDR_ANY interface. If specified, the z/VM TCP/IP server determines the active and available IPv4 TCP/IP interfaces on the system that the LDAP server binds and listens to for requests. See the description of the `bind()` function in the *C/C++ for z/VM Run-time Library Reference* for more information about the **INADDR_ANY** interface.

in6addr_any

Specifies the in6addr_any interface. If specified, the z/VM TCP/IP server determines the active and available IPv4 and IPv6 TCP/IP interfaces on the system that the LDAP server binds and listens to for requests.

portNumber

Specifies the port number. The *portNumber* is optional. If the port number is not specified for an **ldap://**, then the default of 389 is used for nonsecure connections. If the port number is not specified for an **ldaps://**, then the default of 636 is used for secure connections.

The range is 1 to 65536.

It is advisable to reserve the port number or numbers chosen here in PROFILE TCPIP.

pc

Specifies that the LDAP server should listen on the Program Call interface for change log requests from RACF. Only one LDAP server on a system can listen on the Program Call interface. Note that when the **listen** option is initialized to listen for PC calls on the LDAP server, the **listen** parameter must not include an IP address or a host name and you cannot specify **ldaps**.

Following are some examples of how you can specify *ldap_URL*.

- If you specify:

```
ldap://
```


the LDAP server binds and listens on all active and available IPv4 addresses on the system on the nonsecure default port of 389 for incoming client requests. Note this is not the same as `ldap://INADDR_ANY`, which listens specifically on the **INADDR_ANY** interface on the nonsecure default port of 389, or the `ldap://in6addr_any`, which listens specifically on the **in6addr_any** interface on the nonsecure default port of 389.

- If you specify:

```
ldap://us.endicott.ibm.com:489
```

the LDAP server binds and listens on all of the IPv4 and IPv6 addresses associated with host name `us.endicott.ibm.com` on the nonsecure port of 489 for incoming client requests.

- If you specify:

```
ldap://9.130.77.27
```

the LDAP server binds and listens on IPv4 address `9.130.77.27` on the default nonsecure port of 389 for incoming client requests.

- If you specify:

```
ldaps://us.endicott.ibm.com
```

the LDAP server binds and listens for incoming client requests on the IPv4 and IPv6 addresses associated with host name `us.endicott.ibm.com` on the default secure port of 636.

- If you specify:

```
ldaps://9.130.77.27:736
```

the LDAP server binds and listens on IPv4 address `9.130.77.27` on the secure port of 736.

- If you specify:

```
ldap://:489
```

the LDAP server binds and listens on all active and available IPv4 addresses on the system on the nonsecure port of 489 for incoming client requests. Note that this is not the same as `ldap://INADDR_ANY`, which listens specifically on the **INADDR_ANY** interface on the nonsecure port of 489, or `ldap://in6addr_any`, which listens specifically on the **in6addr_any** interface on the nonsecure port of 489.

- If you specify:

```
ldaps://:777
```

the LDAP server binds and listens on all active and available IPv4 addresses on the system on the secure port of 777 for incoming client requests. Note that this is not the same as `ldap://INADDR_ANY:777`, which listens specifically on the **INADDR_ANY** interface on the nonsecure port of 777, or `ldap://in6addr_any:777`, which listens specifically on the **in6addr_any** interface on the nonsecure port of 777.

- If you specify:

```
ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

the LDAP server binds and listens on the IPv6 address `5f1b:df00:ce3e:e200:20:800:2078:e3e3` on the system on the nonsecure port of 389 for incoming client requests.

- If you specify:

```
ldaps://[:ffff:9.130.77.75]:777
```

the LDAP directory server binds and listens on the IPv4 mapped IPv6 address of `::ffff:9.130.77.75` on the system on the secure port of 777 for incoming client requests.

- If you specify:

```
ldap://[::]
```

the LDAP server binds and listens on all active and available IPv4 and IPv6 addresses on the system on the nonsecure default port of 389 for incoming client requests. Note this is not the same as `ldap://INADDR_ANY`, which listens specifically on the **INADDR_ANY** interface on the nonsecure default port of 389, or `ldap://in6addr_any`, which listens specifically on the **in6addr_any** interface on the nonsecure default port of 389.

- If you specify:

```
ldap://:pc
```

the LDAP server binds and listens on the Program Call interface.

Note: The `listen` parameter deprecates the `security`, `port`, and `securePort` options in the configuration file. If there is a `listen` option specified in the configuration file along with either `security`, `port`, or `securePort`, the `listen` option takes precedence over what has been specified for `security`, `port`, or `securePort`. If using an earlier version of the configuration file with `security`, `port`, or `securePort`, the LDAP server is configured to listen on the port numbers specified for `securePort`, `port`, or both depending upon the security setting. However, it is highly recommended that the LDAP server be configured using the `listen` option.

logfile *filename*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the location of the file where the activity log is written when activity logging is enabled. See [“Activity logging” on page 163](#) for more information.

Refer to [“Specifying a Value for Filename” on page 129](#) for information on specifying the *filename*.

The default is `/etc/ldap/gldlog.output`.

logFileFilter *filter*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies a client IP address filter used to determine the activity that is included or excluded from being logged in the activity log file. Client requests originating from IP addresses allowed by the filter are written to the activity log file specified in the **logfile** configuration option.

The only supported activity log filters are ones using the **ibm-filterIP** attribute type to designate the client IPv4 addresses or IPv6 addresses with no brackets that are to be included or excluded from the activity log file. Hostnames and subnet masks are not supported in these filters. For more information, see [“Activity logging” on page 163](#).

The default is `ibm-filterIP=*`

logFileMicroseconds {on | off}

Global	LDBM	SDBM	GDBM	CDBM
X				

Controls all generated log records containing microseconds in their time stamps. This setting cannot be modified by a **LOG** operator **modify** command. The default does not include microseconds in the

time stamps. If **on**, all activity log time stamps include microseconds. If **off**, microseconds are not included.

The default is **off**

The **GLDLOG_MICROSECONDS** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileMsgs {msgs | nomsgs}

Global	LDBM	SDBM	GDBM	CDBM
X				

Controls log records that are generated when messages are created by the LDAP server. The supported options are:

msgs

Messages that are generated by the LDAP server are written to the activity log in addition to the normal target.

noMsgs

Indicates that messages generated by the LDAP server are not written to the activity log.

The default is **noMsgs**

The **GLDLOG_MSG** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileOps {writeOps | allOps | summary | noOps}

Global	LDBM	SDBM	GDBM	CDBM
X				

Controls which operations generate log records. The supported options are:

writeOps

Log records are created for **add**, **delete**, **modify**, **modrdn**, and extended operations.

allOps

Log records are created for **writeops**, **bind**, **search**, **compare**, **abandon**, and **unbind** operations.

summary

Summary statistics are logged every hour. If any logging is collected, summary data is created hourly.

noOps

No log record is created.

The default is **noOps**

The **GLDLOG_OPS** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileRecordType {begin | both | mergedRecord}

Global	LDBM	SDBM	GDBM	CDBM
X				

Controls when log records are generated. The supported options are:

begin

Log records are created at the beginning of requests.

both

Log records are created at the beginning and the end of requests.

mergedRecord

mergedRecord log records are created for all operations that are logged. These records are generated when the operation ends and contain additional information fields that are also provided in the audit log.

The default is **begin**

The **GLDLOG_TIME** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

Note: If set when using the **GLDLOG_TIME** environment variable, the possible values are **time**, **notime**, and **mergedRecord**.

logFileRolloverDirectory *name*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the name of the BFS directory to which the activity log files are archived or rolled over. The name must be a fully-qualified directory path. If the **logfile** configuration option specifies a file that resides in a BFS directory and this option is not specified, the archived or rolled over activity log file is kept in the same directory. For more information about activity log archiving or roll over, see [“Activity logging” on page 163](#).

The default is the directory specified by the **logfile** configuration option.

logFileRolloverSize *nnn[K | M | G]*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the maximum size in bytes of the activity log file. When the maximum size is reached, the activity log file is rolled over or archived. The value *nnn* can be followed by a **K**, **M**, or **G** to indicate kilobytes, megabytes, or gigabytes, in that order, and must be at least 10K (10240) but no larger than:

```
18446744073709551615
18014398509481983K
17592186044415M
17179869183G
```

Specify 0 to disable activity log file archiving or roll over based on size. For more information about activity log archiving or roll over, see [“Activity logging” on page 163](#).

The default is 0.

logFileRolloverTOD *hh:mm*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the time of day when the activity log file is archived or rolled over. Every day at the specified time, the current activity log file is rolled over or archived. The value must be 00:00 - 23:59. Specify a

value outside this range to disable activity log file archiving or roll over based on time of day. For more information about activity log archiving or roll over, see [“Activity logging” on page 163](#).

The default is 24:00.

logFileVersion {0 | 1}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the activity log version. The versions are 0 and 1. See [“Activity logging” on page 163](#) for more information about the activity log version.

The default is 0

masterServer ldap_URL

Global	LDBM	SDBM	GDBM	CDBM
	X			

Specifies for this backend the location of this replica’s master server. There is no required format for the value; however z/VM and z/OS LDAP clients can follow a **masterServer** value only if it is in LDAP URL format. See [listen](#) for a description of LDAP URL format. The presence of this option indicates that this LDAP server is a read-only replica for this backend and receives updates from a master LDAP server. Any other update requests for this backend received directly by the this LDAP server will be redirected to the master server. You must also specify the **masterServerDN** option in this section of the configuration file. The master server must contain all of the suffixes defined for this backend.

The **masterServer** option can be specified multiple times if there are multiple master servers. In this case, the LDAP client will attempt to contact each server in the list until it is able to establish a connection with one of the servers.

In the following example, `myldap.server.com` is the host name and 3389 is the port number of the LDAP URL:

```
masterServer ldap://myldap.server.com:3389
```

In the following example, the IPv6 address of `5f1b:df00:ce3e:e200:20:800:2078:e3e3` is the IP address and 389 is the port number of the LDAP URL.

```
masterServer ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

masterServerDN dn

Global	LDBM	SDBM	GDBM	CDBM
	X			

Specifies the distinguished name (DN) can always make updates to this basic replication read-only replica backend. The value must be in DN format which is described in [Data Model in z/VM: TCP/IP LDAP Administration Guide](#). The presence of this option indicates that this LDAP server is a read-only replica for this backend and receives updates from a master LDAP server using the specified DN. The specified DN is a special entry that is only used when replicating to this read-only replica backend. The DN has unrestricted update, compare, and search access for all entries in the backend on this server, even if the LDAP server is in maintenance mode. When in maintenance mode, only this DN and an LDAP root administrator can access and update the entries in this backend. All other update operations for this backend received by the replica server are redirected to the master server. Care

must be taken when updating this backend to ensure the replica server remains synchronized with the master server.

You must also specify the **masterServer** option in this section of the configuration file. You cannot specify the **peerServerDN** option.

The **masterServerDN** option indicates that basic replication is configured for this backend section. Therefore, the **masterServerDN** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to on in the CDBM backend database section.

It is recommended, though not required, that the DN have the same suffix as one of the **suffix** option values in the configuration file. [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#) describes how to set up your master server DN.

For information on specifying a value for a distinguished name for this option, see [“Specifying a Value for a Distinguished Name” on page 129](#).

masterServerPW string

Global	LDBM	SDBM	GDBM	CDBM
	X			

Specifies the password for the masterServerDN that will be allowed to make updates for this backend. This option is only applicable for a read-only LDAP server. See [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#) for additional information about the master server password.

Note:

1. Use of the masterServerPW configuration option is strongly discouraged in production environments. Instead, specify your masterServerDN as the distinguished name of an existing entry in the directory information tree, including a **userPassword** attribute. This will eliminate passwords from the configuration file.
2. Password policy does not apply to the entry specified in the **masterServerDN** configuration option when the password is specified in the **masterServerPW** configuration option.
3. The **masterServerPW** option indicates basic replication is configured for this backend section. Therefore, the **masterServerPW** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to on in the CDBM backend database section.

maxConnections num-connections

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the maximum number of concurrently connected clients that the LDAP server allows.

The range is 30 to 65535.

The default is 4096.

The LDAP server limits the number of client connections by restricting the number of file and socket descriptors used by the LDAP server. Some of the descriptors are used by the LDAP server for its own file descriptors and passive socket descriptors. The value specified for this option should take into account that the server uses approximately 10 descriptors for internal functions and uses more depending upon the number of additional sockets used as passive sockets for connection attempts by clients.

Setting these limits too high can affect system performance by using too many resources and deprive other functions of their share of the same resources.

nativeAuthSubtree {all | dn}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

Specifies the distinguished name of a subtree where all of its entries are eligible to participate in native authentication. This option can appear multiple times to specify all subtrees that use native authentication. If this option is omitted or is set to **all**, then the entire directory is subject to native authentication. If **useNativeAuth selected** or **all** is not specified, this option is ignored.

For information on specifying a value for a distinguished name for this option, see [“Specifying a Value for a Distinguished Name”](#) on page 129.

The default is **all**.

nativeUpdateAllowed {on | off | reset}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

When set to **on** or **reset**, enables native password or password phrase changes in the security server to occur through a modify request to the LDBM or CDBM backend if the **useNativeAuth selected** or **all** option is specified.

When set to **reset**, this option also allows a bind to the backend to succeed even if the specified native authentication password is expired, as long as the **PasswordPolicy** control is included in the bind request. After the bind, only the special delete-add modification of the bound user's **userPassword** attribute can be performed to reset the native authentication password. Once complete, other LDAP operations can be performed.

This option does not affect the ability to change a native password or password phrase during a bind operation.

Note: LDAP password policy does not apply to entries participating in native authentication.

The default is **off**.

operationsMonitor {ip | ipAny | all}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the search patterns monitored by the LDAP server. The operations monitor supports two types of search patterns, which are **searchStats** and **searchIPStats**. A **searchStats** pattern consists of the search parameters (search base, scope, filter, and attributes to be returned) and status (SUCCESS or FAILURE). A **searchIPStats** pattern consists of the same elements as in the **searchStats** pattern, but also includes the client IP address. If the operations monitor is enabled, LDAP monitors search statistics for the types of search patterns that are configured. For more information about the operations monitor, see [Operations monitor](#) in *z/VM: TCP/IP LDAP Administration Guide*.

If set to **ip**, then only **searchIPStats** patterns are monitored. This option setting is useful in determining if there are any specific clients spamming the LDAP server.

If set to **ipAny**, then only **searchStats** patterns are monitored. This option is useful for evaluating the performance of search patterns.

If set to **all**, the operations monitor will monitor both **searchStats** and **searchIPStats** patterns. Therefore, each search is included in two search patterns, one matching the **searchStats** pattern and one matching the **searchIPStats** pattern.

The default is ipAny.

operationsMonitorSize *num-entries*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the maximum number of search patterns for which the operations monitor gathers statistics. The value must be 0 - 2147483647. A value of 0 indicates that the operations monitor is turned off. When the operations monitor is turned off, the **cn=operations,cn=monitor** entry is not returned on a **cn=monitor** search.

The default is 1000.

pcIdleConnectionTimeout *num-seconds*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the amount of time in seconds that an idle connection remains valid over the LDAP Program Call interface. After the specified time, the connection is considered no longer in use and any resources associated with the connection are released. Idle connections are detected when the LDAP server receives a new Program Call interface connection or a request on an existing Program Call interface connection.

The value must be either 0 or 30 - 2147483647. A value of 0 indicates that an idle connection will remain indefinitely.

The default is 0 (indefinitely).

The recommended value is 0.

pcThreads *num-threads*

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the number of threads to be initialized to handle incoming connections using the LDAP Program Call interface into the LDAP server. No threads are used if the Program Call interface is not active. The value must be in the range of 2 to 2147483647.

The default is 10.

peerServerDN *dn*

Global	LDBM	SDBM	GDBM	CDBM
	X			

Specifies the distinguished name (DN) that can make updates to this basic replication peer replica backend. The value must be in DN format that is described in Data Model in *z/VM: TCP/IP LDAP Administration Guide*. The presence of this option indicates that this LDAP server is a peer replica for this backend, and can receive updates from another peer LDAP server using the specified DN and processing updates received from clients. The specified DN is a special entry that is only used when replicating to this peer replica backend. The DN has unrestricted update, compare, and search access

for all entries in the backend on this server, even if the LDAP server is in maintenance mode. When in maintenance mode, only this DN and an LDAP root administrator can access and update the entries in this backend.

Update operations for this backend received from you bound as **peerServerDN** (or as an LDAP root administrator when in maintenance mode) are performed on the local database and are not sent to any peer and read-only replica servers. When not in maintenance mode, all other update operations for this backend are performed on the local database and are sent to the other peer and read-only replica servers. Update operations from a peer or a master are never replicated. It does not matter if you are in maintenance mode or not. Updates made by an LDAP root administrator are replicated unless the server is in maintenance mode.

You cannot also specify the **masterServerDN** option in this section of the configuration file.

The **peerServerDN** option indicates that basic peer-to-peer replication is configured for this backend section. Therefore, the **peerServerDN** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to on in the CDBM backend database section.

It is recommended, though not required, that the DN have the same suffix as one of the **suffix** option values in the configuration file. [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#) describes how to set up your peer replica DN.

For information on specifying a value for a distinguished name for this option, see [“Specifying a Value for a Distinguished Name” on page 129](#).

peerServerPW string

Global	LDBM	SDBM	GDBM	CDBM
	X			

Specifies the password for the peerServerDN that will be allowed to make updates for this backend. This option is only applicable for a peer replica LDAP server. See [“Establishing the Root Administrator DN and Basic Replication Replica Server DN and Passwords” on page 85](#) for additional information about the peer server password.

Note:

1. Use of the peerServerPW configuration option is strongly discouraged in production environments. Instead, specify your peerServerDN as the distinguished name of an existing entry in the directory information tree, including a **userPassword** attribute, thereby eliminating passwords from the configuration file.

The **peerServerPW** option indicates basic peer-to-peer replication is configured for this backend section. Therefore, the **peerServerPW** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to on in the CDBM backend database section.

2. Password policy does not apply to the entry specified in the **peerServerDN** configuration option when the password is specified in the **peerServerPW** configuration option.

persistentSearch {on | off}

Global	LDBM	SDBM	GDBM	CDBM
	X		X	

Allows or disallows persistent search for changes made to entries in a backend. When **no** is specified, persistent search requests for this backend are rejected. For more information on persistent search, see [PersistentSearch in *z/VM: TCP/IP LDAP Administration Guide*](#).

The default is off.

plugin pluginType pluginName pluginInit [pluginParameters]

Global	LDBM	SDBM	GDBM	CDBM
X				

Defines a plug-in extension to the LDAP server.

- For *pluginType*:
 - Specify **preOperation**, **clientOperation** or **postOperation**. A **preOperation** plug-in is called by the LDAP server before a client request is processed. A **clientOperation** plug-in is called to process a client request. A **postOperation** plug-in is called after a client request is processed. A **clientOperation** plug-in is called when a client request matches a distinguished name suffix or extended operation object identifier registered for the plug-in.
- For *pluginName*:
 - Specify the file name of the shared library (DLL) containing the plug-in code.
- For *pluginInit*:
 - Specify the name of the plug-in initialization routine. This plug-in routine is called by the LDAP server to allow the plug-in to initialize. The plug-in initialization routine will register supported message types, distinguished name suffixes and extended operation object identifiers supported by the plug-in.
- For *pluginParameters*:
 - Optionally, specify plug-in parameters. The plug-in can retrieve these parameters using the **slapi_pblock_get()** routine.

pwCryptCompat {on | off}

Global	LDBM	SDBM	GDBM	CDBM
X	X			X

Specifies whether to use an EBCDIC version or a UTF-8 version of the crypt() algorithm to encrypt passwords when **pwEncryption crypt** is contained in this section of the configuration file. If **on**, the EBCDIC version of the crypt() algorithm is used. If **off**, the UTF-8 version is used. Note that ASCII is a subset of UTF-8. When sharing LDAP directory data between z/VM and an ASCII-based platform, specify **pwCryptCompat off** to ensure that the encrypted value will be the same on both platforms.

The default is on.

pwEncryption {none | crypt | MD5 | SHA | SSHA | DES:keylabel | AES:keylabel}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

Specifies what encryption or hashing method to use when storing the **userPassword** and **ibm-slapdAdminPw** attribute values in the backend of the directory.

none

Specifies no encryption. The **userPassword** and **ibm-slapdAdminPw** attribute values are stored in clear text format. The stored values are prefixed with the tag {none}. The original value, without the tag, is returned for a search request.

crypt

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the crypt() algorithm before they are stored in the directory. The stored values are prefixed with

the tag {crypt}. There are two version of the crypt() algorithm: an EBCDIC-based version and a UTF-8-based version. See the **pwCryptCompat** option and the notes below for information about selecting which version to use. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

MD5

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are encoded by the MD5 hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {MD5}. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

SHA

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the SHA hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {SHA}. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

SSHA

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the Salted SHA (SSHA) hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {SSHA}. The original password value cannot be retrieved in clear text format. The tag and the base64-encoded hashed and salt values are returned for a search request.

SHA224, SHA256, SHA384, SHA512

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the specified SHA-2 hashing algorithm before they are stored in the directory. The stored values are prefixed with the specified tag (for example, {SHA224}). The original password value cannot be retrieved in clear text format. The tag and the base64-encoded hashed value are returned for a search request.

SSHA224, SSHA256, SSHA384, SSHA512

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the specified Salted SHA-2 hashing algorithm before they are stored in the directory. The stored values are prefixed with the specified tag (for example, {SSHA224}). The original password value cannot be retrieved in clear text format. The tag and the base64-encoded hashed and salt values are returned for a search request.

DES:keylabel

Specifies that **userPassword** and **ibm-slapdAdminPw** attributes values are encrypted by the DES algorithm before they are stored in the directory. The stored values are prefixed with the tag '{DES:keylabel}'. The original password value, without the tag, will be returned for a search request. The key label must refer to an entry in the file referenced by the LDAPKEYS FILEDEF statement. See [“Symmetric Encryption Keys” on page 123](#) for more information.

AES:keylabel

Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are encrypted by the AES algorithm using the specified key label before they are stored in the directory. The stored values are prefixed with the tag {AES:keylabel}. The original password value without the tag is returned for a search request. The key label must refer to an entry in the file referenced by the LDAPKEYS FILEDEF statement. See [“Symmetric Encryption Keys” on page 123](#) for more information.

Note:

1. When a password is stored in an LDBM backend, it is prefixed with the appropriate encryption tag so that when a clear text password is sent on an LDAP API simple bind it can be encrypted in that same method for password verification.
2. The crypt() algorithm, implemented across many platforms, accepts only the first eight characters of a password. As a result, any password supplied on a bind or compare operation that matches the first eight characters of a **userPassword** attribute value encrypted with the crypt() algorithm in the directory will match.

3. When the **pwCryptCompat** option is set to **on**, the values encrypted using the crypt algorithm are not portable to other X/Open-conformant systems if the **userPassword** values are unloaded using DS2LDIF with the **-t** command-line parameter and loaded by another platform's load utility. If the **pwCryptCompat** option is set to **off**, the values encrypted using the crypt algorithm are portable to other X/Open-conformant systems if the **userPassword** values are unloaded using DS2LDIF with the **-t** command-line parameter. The output LDIF file from DS2LDIF can then be loaded using another platform's load utility.
4. If a tagged encrypted **userPassword** attribute value is present and is included in an add or modify operation, the attribute value is added as it is with no additional encryption performed on the value even if the **pwEncryption** option is set to a different type of encryption.

The default is **none**.

pwSearchOutput {binary | base64}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the format of MD5 and SHA encrypted **userPassword** and **ibm-slapdAdminPw** attribute values when retrieved on a search request. This option does not affect the retrieval of Salted SHA (SSHA), SHA-2, or Salted SHA-2 hashed **userPassword** and **ibm-slapdAdminPw** attribute values on a search request.

If set to **binary** and a **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the binary hash.

If set to **base64** and a **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the base64-encoded binary hash.

The default is **binary**.

readOnly {on | off}

Global	LDBM	SDBM	GDBM	CDBM
	X	X	X	X

Specifies the ability to modify the database. The LDAP server SMSG BACKEND operator command can be used to change the backend database to read-write or read-only mode while the LDAP server is running. If **readOnly** is turned on, any attempt to use the LDAP server to modify the database fails.

Note:

1. For GDBM, change log entries continue to be created and trimmed (deleted) by the LDAP server even when **readOnly** is **on**.
2. For SDBM, **readOnly** on does not prevent changing a RACF password during a bind operation, using the *currentvalue/newvalue* format. However, it does prevent changing the password by using a modify operation of the **racfpassword** attribute.
3. When LDBM or CDBM is using native authentication, the RACF password can be changed during bind even though **readOnly** on is specified. The RACF password cannot be changed by using the LDBM or CDBM native authentication modify of the **userpassword** attribute.
4. If authenticating or comparing an LDBM or CDBM entry that is subject to password policy in the LDAP server, **readOnly** on does not prevent the password policy operational attributes from being updated in the entry.

The default is **off**.

referral ldap_URL

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the referral to pass back when the target of a client request is not included in any suffix within the LDAP server. It is also known as the default referral. The `referral` option can appear multiple times and should list equivalent servers. There is no required format for the value; however the z/VM and z/OS LDAP clients can follow a referral value only if it is in LDAP URL format. See [listen](#) for a description of LDAP URL format.

In the following example, `myldap.server.com` is the host name and 3389 is the port number of the LDAP directory URL:

```
referral ldap://myldap.server.com:3389
```

In the following example, the IPv6 address `5f1b:df00:ce3e:e200:20:800:2078:e3e3` is the IP address and 389 is the port number of the LDAP URL:

```
referral ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

schemaPath name

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the name of the file directory containing the LDAP schema database and checkpoint files. A fully-qualified directory path must be specified. The schema database file is automatically created during LDAP server initialization if it does not already exist. The LDAP server must have write access to the schema directory.

The default is `/var/ldap/schema`.

schemaReplaceByValue {on | off}

Global	LDBM	SDBM	GDBM	CDBM
X				

Determines the behavior of modify operations with replace values of the schema entry. When **schemaReplaceByValue off** is specified, a modify operation with replace values for an attribute in the schema entry behaves like a normal modify operation: all the values currently in the attribute are replaced by the values specified in the modify operation. When **schemaReplaceByValue on** is specified, individual values in an attribute in the schema entry can be replaced without removing all the other values currently in the attribute. Except in several specific cases, the values of the attribute that are in the initial LDAP server schema cannot be changed or removed. For more information on modifying the schema, see [Updating the Schema in z/VM: TCP/IP LDAP Administration Guide](#).

The **schemaReplaceByValue** configuration option can be overridden on a specific modify operation by including the **schemaReplaceByValueControl** control in the modify request.

The default is **on**.

secretEncryption {none | DES:keylabel | AES:keylabel}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

Specifies the encryption method to use when storing the **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-slappedMasterPw** attribute values in this backend. Applications may use the **secretKey** attribute type to store sensitive data that needs to be encrypted in the directory and to retrieve the data in clear text format. This encryption method is used to protect the **replicaCredentials** attribute values in this backend when basic replication is enabled. This encryption method also protects the **ibm-replicaKeyPw** and **ibm-slappedMasterPw** attribute values in this backend when advanced replication is enabled.

none

Specifies no encryption. The **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-slappedMasterPw** attribute value is stored in clear text format. The stored value is prefixed with the tag {none}. This is the default if the **secretEncryption** option is not specified. The attribute value without the tag is returned for a search request.

DES:keylabel

The **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-slappedMasterPw** attribute value is encrypted by the DES algorithm before it is stored in the directory. The stored value is prefixed with the tag {DES:keylabel}. The original value without the tag is returned for a search request. The key label must refer to an entry in the file referenced by the LDAPKEYS FILEDEF statement. For more information, see [“Symmetric Encryption Keys” on page 123](#).

AES:keylabel

The **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, and **ibm-slappedMasterPw** is encrypted by the AES algorithm before it is stored in the directory. The stored value is prefixed with the tag {AES:keylabel}. The original value without the tag is returned for a search request. The key label must refer to an entry in the file referenced by the LDAPKEYS FILEDEF statement. For more information, see [“Symmetric Encryption Keys” on page 123](#).

The default is none.

securityLabel {on | off}

Global	LDBM	SDBM	GDBM	CDBM
X				

Determines if the security label processing is activated with bound LDAP clients. When **on**, the security labels associated with the LDAP client and LDAP server are verified during the authentication process. Security labels are recorded in all LDAP audit records. When **off**, no security label processing is done.

The default is **off**.

sendV3stringsoverV2as {UTF-8 | ISO8859-1}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the output data format to use when sending **UTF-8** information over LDAP 2.

The default is **UTF-8**.

For more detailed information on the use of this setting, see [UTF-8 data over the LDAP Version 2 protocol in *z/VM: TCP/IP LDAP Administration Guide*](#).

serverEtherAddr mac_address

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the Media Access Control (MAC) address used for entry UUID generation. This value must be unique for all LDAP servers in your enterprise. If multiple LDAP servers run on a (hardware) system, you must specify the MAC address. This applies if your LDAP servers are on different LPARs and also if two LDAP servers are on the same LPAR. You do not need to specify this field if this is the only LDAP server that will run on this (hardware) system.

The MAC address consists of 12 hexadecimal digits. The suggested form of the *mac_address* is:

```
4xmmmmssssss
```

Where:

x

Is a one-character LDAP directory number. If more than one LDAP server is operating on a CPU, specify a different x value for each server. If more than 16 LDAP servers are desired, then use a serial number and model number from a CPU that is not running an LDAP server. If another CPU is not available, then set the x, *mmmm*, and *ssssss* values from the MAC address on an old Ethernet card that is no longer being used or not used to run an LDAP server.

mmmm

Is the four-digit model number of the CPU.

ssssss

Is the six-digit serial number of the CPU.

It is not necessary to follow this convention if you will specify the **serverEtherAddr** option for all LDAP servers in your enterprise. In this case, you can specify any combination of 12 hexadecimal digits as long as each LDAP server has a unique value.

Following is an example:

```
serverEtherAddr 4A123401234D
```

The default is that the LDAP server uses the hardware model and serial numbers to generate a MAC address.

sizeLimit num-limit

Global	LDBM	SDBM	GDBM	CDBM
X	X	X	X	X

Specifies the maximum number of entries to return from a search operation. The maximum number can be modified on a specific search request as described below.

The range is 0 - 2147483647.

0 means "no limit".

The default is 500.

This option applies to all backends, unless specifically overridden in a backend definition or in group search limits. Specifying this before a database line in the configuration file sets the option for all backends. Specifying it after a database line sets the option just for the backend defined by the database line. Specifying a size limit using group search limits sets the limit only for the members of that group. See [Managing group search limits](#) in *z/VM: TCP/IP LDAP Administration Guide* for more information about group search limits.

A limit on the number of entries returned can also be specified by the client on a search request. Note that the following behavior is used when determining the size limit for a search request.

- If the client has not bound as an administrator:
 - If a group search size limit exists for the requestor, then the size used to limit the search is the smaller of the size limit passed by the client and the group search size limit. If the client does not specify a size limit on the search, then the group search size limit is used.

- If a group search size limit does not exist for the requestor, then the size used to limit the search is the smaller than the size limit passed by the client and the size limit read by the server from the **sizeLimit** configuration option in the configuration file (which defaults to 500). If the client does not specify a size limit, then the server size limit is used.
- If the client has bound as an administrator, the size limit is the value passed by the client. If the client does not specify a limit, then the number of entries returned is unlimited. The size limits from the configuration file and from group search limits are ignored when the client has bound as an administrator.

When accessing the LDAP directory support for RACF (the SDBM backend):

- The limit is the smaller of the limit passed by the client and the limit read by the server from the **sizeLimit** option in DS CONF (which defaults to 500). If the client does not specify a limit, then the server limit is used. It does not matter how the client has bound.
- The number of entries returned may be further restricted by limits imposed by RACF. For more information, see [Accessing RACF information in z/VM: TCP/IP LDAP Administration Guide](#).

There are additional considerations for size limit when performing a subtree search from the rootDSE (a NULL-based search). For more information, see [Root DSE search with subtree scope \(Null-based subtree search\)](#) in [z/VM: TCP/IP LDAP Administration Guide](#).

srvStartUpError {terminate | ignore}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies whether the LDAP server stops if a backend fails to initialize after the configuration file is read. If **terminate**, the server ends when any backend fails to initialize. If **ignore**, the LDAP server continues processing as long as the schema successfully initializes. Note that a configuration error that occurs before backend initialization begins always causes the server to end.

The default is **terminate**.

sslAuth {serverAuth | serverClientAuth}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the SSL/TLS authentication method. The **serverAuth** method allows the LDAP client to validate the LDAP server on the initial contact between the client and the server.

The **serverClientAuth** method allows the LDAP client to validate the LDAP server. In addition, the LDAP server validates the LDAP client if the client sends its digital certificate on the initial contact between the client and the server.

Note: In order for clients to perform **SASL EXTERNAL** binds to the LDAP server, it is necessary to configure the server with **sslAuth serverClientAuth**.

For more SSL/TLS information, see [“Setting up for SSL/TLS” on page 114](#).

The default is **serverAuth**.

sslCertificate {certificateLabel | none}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the label of the certificate that is used for LDAP server authentication. The certificate is stored in the key database file which is created and managed using the GSKKYM utility. For details on using the GSKKYM utility, see [SSL Certificate Management in z/VM: TCP/IP User's Guide](#). For more SSL/TLS information, see [“Setting up for SSL/TLS” on page 114](#).

The default is none.

If the value is none (by default or by specification), the default certificate used for server authentication is the one marked in the key database file managed by the GSKKYM utility.

sslCipherSpecs {string | GSK_V3_CIPHER_SPECS_EXPANDED | ANY}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the SSL 3.0 and TLS 1.0 cipher specifications that the LDAP server accepts from clients. Use of this option to specify the specific cipher suites is limited, and provided only for compatibility with earlier versions. It supports only a portion of the cipher suites available in System SSL, contains no 4-character cipher suites, and provides no order of preference. The preferred approach is to set the option to GSK_V3_CIPHER_SPECS_EXPANDED and then set the environment variable GSK_V3_CIPHER_SPECS_EXPANDED to the list of 4-character cipher specifications you want, in order of preference.

If the cipher specifications you want are included in [Table 22 on page 118](#) and if the order of preference matches the default order that is provided by System SSL, then the **sslCipherSpecs** option may be used with any of the values that are described.

In this case, the cipher specification is a blank delimited string that represents an ORed bit-mask indicating the SSL/TLS cipher specifications that are accepted from clients. Clients that support any of the specified cipher specifications are able to establish an SSL/TLS connection with the server. [Table 22 on page 118](#) lists the CipherSpec mask values and the related decimal, hexadecimal, and keyword values. See *z/OS Cryptographic Services System SSL Programming* for a description of supported cipher specifications.

The cipher specification may be specified as follows:

- A decimal value (for example, 256)
- A hexadecimal value (for example, x100)
- A keyword (for example, **TRIPLE_DES_SHA_US**)
- A construct of those values using plus and minus signs to indicate inclusion or exclusion of a value. For example,
 - 256+512 is the same as specifying 768, or x100+x200, or TRIPLE_DES_SHA_US+DES_SHA_EXPORT
 - 52992 is the same as specifying ALL-RC2_MD5_EXPORT-RC4_MD5_EXPORT

Depending upon the level of SSL support installed, some ciphers may not be supported. SSL will ignore the unsupported ciphers. You should consult the SSL documentation to determine the specific ciphers that your installation supports.

For more SSL/TLS information, see [“Setting up for SSL/TLS” on page 114](#).

The default is xCCC000.

The default value includes the following ciphers:

Cipher name	SSL value	Hexadecimal value
RSA_AES_256_SHA	35	x00008000
DHE_DSS_AES_256_SHA	38	x00040000

Cipher name	SSL value	Hexadecimal value
DHE_RSA_AES_256_SHA	39	x00080000
RSA_AES_128_SHA	2F	x00004000
DHE_DSS_AES_128_SHA	32	x00400000
DHE_RSA_AES_128_SHA	33	x00800000

sslKeyRingFile name

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies either the path and file name of the SSL/TLS key database file for the LDAP server. SSL/TLS connections will not be available if this option is not specified.

The file name must match the key database file name that was created using the GSKKMAN utility (see *SSL Certificate Management in z/VM: TCP/IP User's Guide*). Also, for more SSL/TLS information, see “Setting up for SSL/TLS” on page 114.

sslKeyRingFilePW string

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies the password protecting access to the SSL/TLS key database file. The password string must match the password to the key database file that was created using the GSKKMAN utility (see *SSL Certificate Management in z/VM: TCP/IP User's Guide*). Also, see “Setting up for SSL/TLS” on page 114 for more SSL/TLS information.

Note: Use of the sslKeyRingFilePW configuration option is strongly discouraged. As an alternative, use the sslKeyRingPWStashFile configuration option. This will eliminate this password from the configuration file.

sslKeyRingPWStashFile name

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies a file system file name where the password for the server's key database file is stashed. Use the full path name of the stash file in the file system for *name*.

If this option is present, then the password from this stash file overrides the sslKeyRingFilePW configuration option, if present. Use the GSKKMAN utility with the -s option to create a key database password stash file. For more SSL/TLS information, see “Setting up for SSL/TLS” on page 114.

suffix dn_suffix

Global	LDBM	SDBM	GDBM	CDBM
	X	X		

Denotes the distinguished name of the root of a subtree in the namespace managed by this backend within the LDAP server. This option may be specified more than once to indicate all the roots of the subtrees within this backend except for the SDBM backend. The SDBM backend must have only one suffix. Note that a suffix cannot be specified for the GDBM, CDBM, and EXOP backends. When the GDBM backend is configured, the **cn=changelog** suffix is reserved. When the CDBM backend is configured, the **cn=configuration** and **cn=ibmpolicies** suffixes are reserved. The special suffix,

cn=localhost, can be placed in any LDBM backend and is exempt from replication when advanced replication is used.

Identical and overlapping suffixes cannot be specified in the LDAP server configuration file, even if the suffixes are within different backends. These suffixes create confusion and can result in unexpected results. An example of overlapping suffixes is:

```
suffix ou=Server Group, o=IBM
suffix o=IBM
```

For information about specifying special characters and undefined attributes in the suffix, see [“Specifying a Value for a Distinguished Name” on page 129](#).

Domain Component naming as specified by RFC 2247 is also supported in the LDAP server. For example, the domain name `ibm.com` could be specified as the following suffix in the configuration file:

```
suffix "dc=ibm,dc=com"
```

tcpTerminate {terminate | recover}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies whether the LDAP server ends when network interfaces are not active. The LDAP server periodically polls the network interfaces it is using to determine when they go down and come back up. If an interface fails but the LDAP server still has at least one active interface, the server continues processing and reestablishes a failed interface when it detects that it has become active. If all interfaces fail and **tcpTerminate terminate** is specified, the LDAP server ends. If **tcpTerminate recover** is specified, then the LDAP server remains active and attempts to reestablish network interfaces when it detects they have become active. All client operations targeted to the LDAP server fail until a network interface can be reconnected. The frequency of polling can be set using the **LDAP_NETWORK_POLL** environment variable. For more information, see [“Step 8. Set Environment Variables \(DS ENVVARS\)” on page 105](#).

The **tcpTerminate** option is also used to determine whether the LDAP server ends if SSL initialization fails during server initialization. If **terminate** is specified, the LDAP server ends. If **recover** is specified, the LDAP server continues initialization, but the failed SSL interface cannot be used until the error is fixed and the LDAP server is restarted.

The default is `recover`.

timeLimit num-seconds

Global	LDBM	SDBM	GDBM	CDBM
X	X	X	X	X

Specifies the maximum number of seconds (in real time) the LDAP server will spend answering a search request. This maximum number can be modified on a specific search request as described below. If a request cannot be processed within this time, a result indicating an exceeded time limit is returned.

The range is 0 - 2147483647.

0 means "no limit".

The default is 3600.

This option applies to all backends, unless specifically overridden in a backend definition or in group search limits. Specifying this before a database line in the configuration file sets the option for all backends. Specifying it after a database line sets the option just for the backend defined by the

database line. Specifying a time limit using group search limits sets the limit only for the members of that group. See [Managing group search limits](#) in *z/VM: TCP/IP LDAP Administration Guide* for more information about group search limits.

A limit on the amount of time can also be specified by the client on a search request. Note that the following behavior is used when determining the time limit for a search request.

- If the client has not bound as an administrator:
 - If a group search time limit exists for the requestor, then the time used to limit the search is the smaller of the time limit passed by the client and the group search time limit. If the client does not specify a time limit on the search, then the group search time limit is used.
 - If a group search time limit does not exist for the requestor, then the time used to limit the search is the smaller than the time limit passed by the client and the time limit read by the server from the **timeLimit** configuration option in the configuration file (which defaults to 500). If the client does not specify a time limit, then the server time limit is used.
- If the client has bound as an administrator, the time limit is the value passed by the client. If the client does not specify a limit, then the amount of time is unlimited. The time limits from the configuration file and from group search limits are ignored when the client has bound as an administrator.

When accessing the LDAP support for RACF (the SDBM backend):

- The limit is the smaller of the limit passed by the client and the limit read by the server from the **timeLimit** option in DS CONF (which defaults to 3600). If the client does not specify a limit, then the server limit is used. It does not matter how the client has bound.

There are additional considerations for time limit when performing a subtree search from the rootDSE (a NULL-based search). For more information, see [Root DSE search with subtree scope \(Null-based subtree search\)](#) in *z/VM: TCP/IP LDAP Administration Guide*.

useAdvancedReplication {on | off}

Global	LDBM	SDBM	GDBM	CDBM
				X

Specifies whether the LDAP server supports advanced replication. If advanced replication is active, then the **masterServer**, **masterServerDN**, **masterServerPW**, **peerServer**, **peerServerDN**, and **peerServerPW** configuration options cannot be specified in any LDBM or CDBM backends.

Note:

1. The LDAP server does not start when **useAdvancedReplication** on is specified and entries with an objectclass of **replicaObject** are present in an LDBM or CDBM backend. If entries with an objectclass of **replicaObject** are attempted to be added or modified in this configuration, the add or modify request is rejected.
2. The LDAP server does not start when **useAdvancedReplication** off is specified and entries with an objectclass of **ibm-replicationAgreement**, **ibm-replicationContext**, **ibm-replicationGroup**, or **ibm-replicationSubEntry** are present in an LDBM or CDBM backend. If entries with these objectclass values are attempted to be added or modified in this configuration, the add or modify request is rejected.

For additional information about advanced replication, see [Advanced replication](#) in *z/VM: TCP/IP LDAP Administration Guide*.

The default is off.

useNativeAuth {selected | all | off}

Global	LDBM	SDBM	GDBM	CDBM
	X			X

Enables native authentication in the backend. If the value is:

- **selected**, only entries with the **ibm-nativeId** attribute that are within the native subtrees (see [nativeAuthSubtree](#)) use native authentication.
- **all**, all entries within native subtrees use native authentication. These entries can contain the **ibm-nativeId** or **uid** attribute to specify the RACF ID.
- **off**, no entries participate in native authentication.

Note: LDAP password policy does not apply to entries participating in native authentication.

The default is **off**.

validateincomingV2strings {on | yes | off | no}

Global	LDBM	SDBM	GDBM	CDBM
X				

Specifies whether the incoming strings are validated. If set to on, this setting limits the format of incoming string data sent over the LDAP 2 to the IA5 character set (X'00'-X'7F' or "7-bit ASCII"). With this setting, textual data received on operations outside of the IA5 character set causes the operations to fail with LDAP_PROTOCOL_ERROR.

The default is **on**.

Note that while supported, it is not recommended to run with this data filtering disabled.

Dynamic Server Operation

The VM Special Message Facility (MSG) command provides an interactive interface to the LDAP virtual machine to perform privileged system administration tasks. Privileged users are specified in the OBEY list of the TCP/IP server configuration file.

SMSG Interface to the LDAP Server

```
➤ SMSG — server_id — cmd-name — cmd-options ➤
```

Purpose

Use the VM Special Message Facility (MSG) interface to the LDAP virtual machine to:

- Control auditing
- Change a backend to read-write or read-only mode
- Force all file-based backends to commit their changes
- Set debugging levels
- Display information about the LDAP server
- Turn logging on or off
- Change the LDAP server between normal and maintenance mode

- Initialize the SSL environment
- Reset various counters maintained by the LDAP server
- Stop the LDAP server

Operands

server_id

Specifies the user ID of the LDAP server virtual machine.

The list below describes the supported LDAP server operator commands (*cmd-name*) and their command options (*cmd-options*).

AUDIT audit_controls

Turn LDAP server auditing on or off and control what server activities result in creating an audit record. For more information on controlling LDAP server's usage of audit, see [“LDAP SMF Auditing” on page 168](#).

BACKEND backendName=RDWR | READ

Change a specific backend to read-write (normal) mode or read-only mode. The LDAP server can place a file-based backend into read-only mode if the LDAP server cannot access its checkpoint or database files. After correcting the access problem, the operator can use this command to reset the backend to read-write mode.

COMMIT

Force all the file-based backends to commit their changes to the database files by merging in the changes from the checkpoint files. The changes are removed from the checkpoint files. This can be done to prevent the checkpoint files from growing too large.

DEBUG level DEBUG OUTPUT=MEMORY | BOTH

Set the level of debugging. For more information on setting the debug levels, see [“Dynamic Debugging” on page 163](#). The command can also control whether debug output is sent to just the internal table only or to the internal table and the normal debug output destination.

DISPLAY AUDIT | BACKENDS | DEBUG | LEVEL | LOCKS | MAINTMODE | MONITOR | NETWORK | REPLICAS | THREADS

Display a variety of information about the LDAP server.

LOG WRITEOPS | ALLOPS | SUMMARY | TIME | NOTIME | MERGEDRECORD | MSGS | NOMSGS | FLUSH | STOP | ROLLOVER | FILTER, filter

Turn LDAP server activity logging on or off and control what server activities are logged. For more information, see [“Activity logging” on page 163](#).

MAINTMODE ON | OFF

Change the LDAP server between normal mode and maintenance mode. Access to the LDAP server is restricted to certain users when in maintenance mode, so this mode can be used to fix the LDAP server. For more information about basic replication maintenance mode, see [“Basic replication maintenance mode” in z/VM: TCP/IP LDAP Administration Guide](#). For more information about advanced replication mode, see [“Advanced replication maintenance mode” in z/VM: TCP/IP LDAP Administration Guide](#).

REFRESH SSL

Initialize the SSL environment again. This may be necessary, for example, if SSL replaces expired certificates.

RESET LOCKS | MONITOR | THREADS

Reset various counters that the LDAP server maintains.

SHUTDOWN

Stop the LDAP server.

UNLOCK ADMIN

Unlocks the LDAP root administrator entry in an LDBM or CDBM backend when the password has expired or the maximum number of failed bind attempts in the effective password policy has been exceeded.

After successful completion of this command, the LDAP root administrator must change his or her password before authenticating to the LDAP server. This command is only valid for unlocking the LDAP root administrator that is specified in the configuration file (**adminDN** configuration option) and exists as an LDBM or CDBM backend entry with a user **userPassword** value. This command is not valid when the LDAP root administrator's entry is participating in native authentication, resides in the SDBM backend, or the password is specified in the configuration file in the **adminPW** configuration option.

Dynamic Debugging

When the LDAP server is running it is possible to dynamically turn the debugging facility on and off. You can also replace the current debug levels, add to the current debug levels, or remove from the current debug levels. The following command can be sent to the LDAP server from the CMS. In the command:

```
smsg ldapsrv debug debug_level
```

The *debug_level* is a mask that specifies the desired debug level. See [Table 21 on page 108](#) for an explanation of the debug level values.

Debug information will be added to the output associated with the LDAP server.

To turn the debug tracing off, enter the same command providing the value zero (0) or OFF for *debug_level*.

Activity logging

The LDAP server supports logging client activity in an activity log file. The activity log file can be analyzed for LDAP server load analysis to determine the client operations handled by the server. The activity log records can contain information about operations handled by the server, client IP addresses, messages generated by the server, and hourly activity summary statistics. The LDAP server activity logging support has a number of features that allow the LDAP administrator to customize the client activity to be logged.

The available versions of activity logging are version 0 and version 1. Activity log version 1 is the enhanced activity logging version that includes more features to help further debugging LDAP issues and inappropriate user operations of the LDAP server besides all features in version 0. For example, the addition of logging attributes in the add and modify requests allows you to detect when and who modified the critical attributes. The connect and disconnect records allow the detection of software configuration errors or denial of service attacks. The abandon record shows when a request is abandoned and the msgid of the request that is abandoned.

The features for version 0 include:

- Logging the start or end of a client operation
- Logging only client update operations (add, delete, modify, extended operations, and modifydn)
- Logging all client operations (add, bind, compare, delete, extended operations, modify, modifydn, search, and unbind)
- Logging messages generated by the server
- Logging hourly client activity summary statistics
- Logging only requests from certain client IP addresses
- Activity log file archiving or rollover which copies the current activity log file to another location for load analysis.

The features for version 1 include:

- Logging the start or end of a client operation.
- Logging only client update operations (add, delete, modify, extended operations, and modifydn).
- Logging all client operations (add, bind, compare, delete, extended operations, modify, modifydn, search, and unbind).
- Logging messages that are generated by the server.

- Logging hourly client activity summary statistics.
- Logging only requests from certain client IP addresses.
- Activity log file archiving or rollover that copies the current activity log file to another location for load analysis.
- A configuration keyword that enables the new version, thus users that want the previous behavior are not effected.
- Logging the start and end of a connection.
- Logging abandon requests.
- Adding the msgid to requests that can be abandoned so that abandon requests can be matched to the requests they affect. The requests include: add, bind, compare, delete, exop, modify, rename, and search.
- Logging the attribute names in the add request records.
- Logging the attribute names and an indication of the attribute being added, deleted, or replaced in the modify request records.
- Logging requests unknown to the activity log.

Start and end connection logging

The following is an example of version 1 merged activity log records containing connection start and connection end messages:

```
Thu Oct 8 08:37:59 2009 mergedRecord Connect: connid = 1374, clientIP = 1.2.3.4,
listen = ldap://[fe00::f4f7:0:0:7442:7510]:389
Thu Oct 8 08:50:59 2009 mergedRecord Disconnect: connid = 1374, clientIP = 1.2.3.4,
bind = 'cn=john', cause = 1
```

The following is an example of version 1 non-merged activity log records containing connection start and connection end messages:

```
Thu Oct 8 08:43:43.424715 2009 Connect: connid = 1, listen =
ldap://[fe00::f4f7:0:0:7442:7510]:389, IP = 1.2.3.4,
Thu Oct 8 08:49:45.424716 2009 Disconnect: connid = 1, DN = 'cn=john', cause = 1,
IP = 1.2.3.4
```

Abandon request

The following is an example of a version 1 merged activity log record for an abandon request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Abandon : connid = 1374,
clientIP = 9.12.47.67, time = 711usec, msgid = 123, targetmsgid = 456
```

The following is an example of version 1 non-merged activity log records for an abandon request:

```
Thu Oct 8 08:43:43.424715 2009 Abandon: connid = 1, msgid = 123, targetmsgid = 456,
IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Abandon: connid = 1, msgid = 123,
targetmsgid = 456, IP = 1.2.3.4
```

Attribute names in add request records

Attribute names is a string of attribute names that are separated by spaces. The following is an example of a version 1 merged activity log record for an add request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Add: connid = 1374, clientIP = 1.2.3.4,
bind = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, time = 49268usec,
controls = 1.3.6.1.4.1.42.2.27.8.5.1, target = ou=zos,o=ibm,c=us, msgid = 789,
attrs = cn objectclass description, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for an add request:


```
Thu Oct 8 08:43:43.424715 2009 Add: connid = 1, DN = 'cn=john,ou=zos,o=ibm,c=us', msgid = 789,
IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Add: connid = 1, DN = cn=john,ou=zos,o=ibm,c=us, rc = 0,
msgid = 789, attrs = cn objectclass description, IP = 1.2.3.4
```

Attribute names in modify request records

Version 1 of activity logging provides attribute names and an indication of the attribute being added, deleted, or replaced in the modify request records.

Attribute names in a modify request record is a string with an indication of:

- the action taken separated by spaces, ended by a comma.

- "-" indicates that the attribute is deleted.

- "+" indicates that the attribute is added.

- !" indicates that the attribute is replaced.

The following is an example of a version 1 merged activity log record for a modify request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Modify: connid = 1374, clientIP = 1.2.3.4,
bind = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, time = 49268usec,
controls = 1.3.6.1.4.1.42.2.27.8.5.1, target = 'ou=zos,o=ibm,c=us', msgid = 789,
attrs = -telephone +mobilephone !socsecuritynumber, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for a modify request:

```
Thu Oct 8 08:43:43.424715 2009 Modify: connid = 1, DN = 'cn=john,ou=zos,o=ibm,c=us', msgid =
789,
IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Modify: connid = 1,
DN = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, msgid = 789, attrs = -telephone +mobilephone
!socsecuritynumber, IP = 1.2.3.4
```

Logging requests unknown to activity logging

The following is an example of a version 1 merged activity log record for an unknown request:

```
Thu Oct 8 08:37:59.424715 2009 mergedRecord Unknown: type = 25, connid = 1, clientIP =
9.12.47.67,
bind = 'cn=Admin', rc = 0, time = 711usec, controls = , msgid = 2, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for an unknown request:

```
Tue Oct 9 02:16:23.822710 2009 Unknown: type = 25, connid = 1, msgid = 2, IP = 9.12.47.67
Tue Oct 9 02:16:23.822789 2009 End Unknown: type = 25, connid = 1, msgid = 2, IP = 9.12.47.67
```

Configuring the activity log support

The **logfile** configuration option in the global section of the LDAP server configuration file specifies the location of the BFS or CMS file where activity log records are written. A fully qualified name must be specified.

The following is an example of a **logfile** configuration option that specifies a BFS file:

```
logfile /etc/ldap/ldap.activity.log
```

If the **logfile** configuration option is not specified, the default location of the activity log file is **/etc/ldap/gldlog.output**.

A CMS file can be specified in the **logfile** configuration option by using a ddname or by specifying a specific dataset. The following is an example of both methods.

```
logfile //dd:logout
logfile //ldap.actlog.a
```

For more information about the **logfile** configuration option, see [logfile](#).

The LDAP server supports automatic activity log file rollover or archiving based on the time of day or the size of the log file. The activity log archiving is supported only when the log file is a BFS file. When the **logFileRolloverTOD** configuration option has a value 00:00 - 23:59, it indicates the time each day when the current activity log file is archived. For more information about the **logFileRolloverTOD** configuration option, see [logFileRolloverTOD](#). When the **logFileRolloverSize** configuration option has a non-zero size, it indicates the size in bytes, megabytes, kilobytes, or gigabytes that the activity log file is required to have before it is archived or rolled over. For more information about the **logFileRolloverSize** configuration option, see [logFileRolloverSize](#). When the activity log file reaches one of these thresholds or is manually rolled over with the SMSG LOG command, the following occurs:

- The current activity log file is renamed with the current Zulu time stamp appended to the end of the file name.
- If the **logFileRolloverDirectory** configuration option is specified, the archived log file is moved to that directory, or else the archived activity log file is left in the same directory as the current activity log file. For more information about the **logFileRolloverDirectory** configuration option, see [logFileRolloverDirectory](#).
- When archiving is complete, the server opens a new activity log file with the name specified in the **logfile** configuration option.

When the activity log is configured to log client operations and is active, the LDAP server logs all client operations from all IP addresses. However, the activity log can be configured to include or exclude client operations from certain IP addresses by using the **logFileFilter** configuration option or the SMSG LOG command. For more information about the **logFileFilter** configuration option, see [logFileFilter](#). An IETF RFC 2254 (*The String Representation of LDAP Search Filters*) compliant LDAP search filter can be specified in the **logFileFilter** configuration option or the SMSG LOG command using only the **ibm-filterIP** attribute type in the filters. The following example **logFileFilter** configuration option specifies to only log client operations from IP addresses starting with 1.2.3* or 1.2.4*.

```
logFileFilter (|(ibm-filterIP=1.2.3*)(ibm-filterIP=1.2.4*))
```

The default data collection setting is to collect no data in the activity log. The default is modified by environment variables or through the usage of the LDAP server SMSG LOG command. The environment variables are read as the server starts up and the SMSG LOG command can be specified once the server has started. The **logFileFilter** configuration option can also be updated with the SMSG LOG command.

To enable the version 1 features of the activity log, specify the **logFileVersion** configuration option in the configuration file, with the value of 1. For more information about how to configure the version, see “Configuration File Options” on page 132 for the **logFileVersion** configuration option.

The syntax of the SMSG ldapsrv LOG command is:

```
SMSG ldapsrv LOG WRITEOPS | ALLOPS | SUMMARY | TIME | NOTIME | MERGEDRECORD |  
MSG | NOMSGS | FLUSH | STOP | ROLLOVER | FILTER,filter
```

If the activity log must be restarted with logging the beginning and end of all operations, specify the following LOG operator modify commands:

```
SMSG ldapsrv LOG ALLOPS
```

then,

```
SMSG ldapsrv LOG TIME
```

The **logFileOps** configuration option and the **LOG** operator modify command can be used to control which operations generate log records. Before specifying an operation setting, no operation logging is performed. See the **logFileOps** configuration option at “Configuration File Options” on page 132 for more information.

Summary records are created on an hourly basis, when rollover or archiving occurs, or when an LDAP server LOG operator modify command is processed.

Note: If activity logging is not active, summary records are not created.

The summary log records contain information about the operations that the server has processed.

Following is an example of the summary log records:

```
Tue Jun 15 14:26:43.785091 2010 total operations started = 113780
Tue Jun 15 14:26:43.785165 2010 total operations completed = 113780
Tue Jun 15 14:26:43.785185 2010 total binds completed = 31926
Tue Jun 15 14:26:43.785202 2010 total unbinds completed = 31925
Tue Jun 15 14:26:43.785221 2010 total searches completed = 17092
Tue Jun 15 14:26:43.785238 2010 total modifies completed = 14058
Tue Jun 15 14:26:43.785254 2010 total adds completed = 7013
Tue Jun 15 14:26:43.785270 2010 total deletes completed = 7013
Tue Jun 15 14:26:43.785286 2010 total modifydns completed = 1290
Tue Jun 15 14:26:43.785302 2010 total compares completed = 133
Tue Jun 15 14:26:43.785318 2010 total abandons completed = 0
Tue Jun 15 14:26:43.785334 2010 total extendedops completed = 3330
Tue Jun 15 14:26:43.785349 2010 total unknown completed = 0
Tue Jun 15 14:26:43.785355 2010 total group gatherings completed = 98876
Tue Jun 15 14:26:43.785366 2010 total search entries sent = 332904
Tue Jun 15 14:26:43.785381 2010 total bytes sent = 83800442
Tue Jun 15 14:26:43.785396 2010 total search references sent = 0
Tue Jun 15 14:26:43.785411 2010 total search pages sent = 1688
Tue Jun 15 14:26:43.785427 2010 total paged searches completed = 844
Tue Jun 15 14:26:43.785445 2010 total sorted searches completed = 3423
Tue Jun 15 14:26:43.785459 2010 total connections processed = 31926
Tue Jun 15 14:26:43.785476 2010 current connections = 0
Tue Jun 15 14:26:43.785491 2010 connection high water mark = 2
Tue Jun 15 14:26:43.785506 2010 connections timed out = 0
Tue Jun 15 14:26:43.785521 2010 paged result sets timed out = 0
```

The **logFileRecordType** configuration option and the **LOG** operator modify command can be used to control when log records are generated. See the **logFileRecordType** configuration option at [“Configuration File Options” on page 132](#) for more information.

The following are examples of activity log records containing start and end records:

```
Thu Oct 8 07:41:08 2009 Bind SIMPLE: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct 8 07:41:08 2009 End Bind SIMPLE: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, safid = , rc = 0,
IP = 1.2.3.4, policyUpdated = T
Thu Oct 8 07:41:08 2009 Search: connid = A, base = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
attrs = , IP = 1.2.3.4, searchFlags = 0
Thu Oct 8 07:41:08 2009 End Search: connid = A, base = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope =
2,
count = 4, rc = 0, IP = 1.2.3.4, searchFlags = 0
Thu Oct 8 07:41:08 2009 Unbind: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct 8 07:41:08 2009 End Unbind: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
```

The following are examples of activity log records merged records:

```
Thu Oct 8 08:37:59 2009 mergedRecord Bind: connid = 1374, clientIP = 1.2.3.4, bind =
cn=john,ou=zos,o=ibm,c=us,
rc = 0, time = 49268usec, controls = 1.3.6.1.4.1.42.2.27.8.5.1, listen = ldap://
[fe00::f4f7:0:0:7442:7510]:389, seclabel = , mech = SIMPLE, saf = , policyUpdated = T, rs=NA
Thu Oct 8 08:37:59 2009 mergedRecord Search: connid = 1374, clientIP = 1.2.3.4, bind =
cn=john,ou=zos,o=ibm,c=us,
rc = 0, time = 1437usec, controls = , target = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
attrs = , count = 4, searchFlags = 0, rs=NA
Thu Oct 8 08:37:59 2009 mergedRecord Unbind: connid = 1374, clientIP = 1.2.3.4, bind =
cn=john,ou=zos,o=ibm,c=us,
rc = 0, time = 6usec, controls = , listen = ldap://[fe00::f4f7:0:0:7442:7510]:389, seclabel = , saf = ,
rs=NA
```

For a description of the different fields that are present in activity log records, see [Appendix C, “Activity Log Records,” on page 707](#).

Note: When activity occurs on the Program Call interface, the IP address is reported as 'PC'.

The **logFileMicroseconds** configuration option controls if all generated log records contain microseconds in their time stamps. This setting cannot be modified by a **LOG** operator modify command. The default does not include microseconds in the time stamps. See the **logFileMicroseconds** configuration option at [“Configuration File Options” on page 132](#) for more information.

The following are examples of activity log records containing microseconds:

```
Thu Oct 8 08:43:43.424715 2009 Bind SIMPLE: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct 8 08:43:43.455011 2009 End Bind SIMPLE: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, safid = , rc = 0, IP = 1.2.3.4,
policyUpdated = F
Thu Oct 8 08:43:43.466558 2009 Search: connid = 1, base = cn=tom,ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
attrs = , IP = 1.2.3.4, searchFlags = 0
Thu Oct 8 08:43:43.471836 2009 End Search: connid = 1, base = cn=tom,ou=zos,o=ibm,c=us, filter = (objectclass=*),
scope = 2,
count = 1, rc = 0, IP = 1.2.3.4, searchFlags = 0
Thu Oct 8 08:43:43.478053 2009 Unbind: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct 8 08:43:43.478181 2009 End Unbind: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
```

The **logFileMsgs** configuration option and the **LOG** operator modify command can be used to control if log records are generated when messages are created by the LDAP server. See the **logFileMsgs** configuration option at [“Configuration File Options”](#) on page 132 for more information.

The following is an example of an activity log record containing a message:

```
Thu Oct 8 08:43:24.748429 2009 GLD1059I Listening for requests on 127.0.0.1 port 389.
```

The activity log filter specified in the **logFileFilter** configuration option can be updated by issuing an SMSG LOG command. The following command updates the server to only log client requests originating from IP address 1.2.4.5.

```
smsg ldapsrv log filter,(ibm-filterip=1.2.4.5)
```

The current activity log can be manually rolled over or archived using the process described above by issuing the following SMSG LOG command:

```
smsg ldapsrv log rollover
```

As the log records are produced, some buffering of the output is performed by the system. The buffers are flushed before the server shuts down. However, you can force the server to flush the buffers by issuing the following SMSG LOG command:

```
smsg ldapsrv log flush
```

To have the server stop collecting activity data, issue the following SMSG LOG command:

```
smsg ldapsrv log stop
```

Activity logging can be started again by specifying an SMSG LOG command with a new setting.

```
smsg ldapsrv log allops
```

The current activity log settings can be queried by issuing the following SMSG DISPLAY command:

```
smsg ldapsrv display log

GLD1290I Activity log status
Option:      Setting
OPERATIONS   ALLOPS
TIME         MERGEDRECORD
MESSAGES     MSGS
MICROSECONDS NOMICRO
FILTER       NONE
```

LDAP SMF Auditing

The LDAP server can be configured to generate SMF type-83 subtype 3 audit records. The SMF type-83 log records containing LDAP events can be unloaded by using the RACF SMF Data Unload utility for further analysis by auditing tools. These audit records contain information provided on LDAP client operation requests. The LDAP server is configured to write audit records when the operation successfully completes, when the operation fails or for either case. SMF type-83 subtype 3 audit records are not

created by the LDAP server when a request is handled by a plug-in. The LDAP server uses RACF to write the record to SMF. For setup information, see [“Additional Setup for Auditing” on page 83](#).

Auditing Events

Auditing of LDAP operations can be set up by using the **audit** option in the LDAP server configuration file. For more information, see [audit](#).

While the LDAP server is running, auditing can be turned *on* or *off* and the specifications of which operations are to be audited and their associated audit level can be changed using the LDAP server AUDIT operator SMSG command. The format of the AUDIT operator SMSG command is:

```
smsg ldapsrv audit on | off | all,operations | error,operations | none,operations
```

When auditing is *on*, an LDAP SMF type 83 subtype 3 audit record is generated for an operation if the operation is specified on an audit level and the operation result matches the audit level.

A separate **audit** configuration option or AUDIT operator SMSG command must be issued to turn auditing *on* or *off* and to set each audit level. Multiple operations can be specified for a level by either putting a *+* between them on the **audit** option or AUDIT command, or by specifying multiple **audit** options or AUDIT commands with the same level.

Operations can be audited only when they fail or all the time. The following audit levels are supported:

all

An LDAP audit record will be generated for the specified operations.

error

An LDAP audit record will be generated for the specified operations when they fail.

none

An LDAP audit record is not generated for the specified operations.

The supported values for operations can be one or more of:

abandon
add
bind
compare
connect
delete
disconnect
exop
modify
modifydn
search
unbind

If an operation is specified in more than one level, the last level is used for the operation. If an operation is not specified in any level, the level defaults to *none* for that operation. Turning auditing *off* does not change the setting for the audit levels. If auditing is later turned *on*, the audit levels will remain as they last were.

For example, the following AUDIT operator SMSG commands turn auditing *on* for modify and search operation failures and for all bind operations. The other operations are not audited.

```
smsg ldapsrv audit error,modify+search+bind  
smsg ldapsrv audit all,bind  
smsg ldapsrv audit on
```

The current audit settings can be displayed using the following LDAP server DISPLAY operator SMSG command:

```
smsg ldapsrv display audit
```

The results for the AUDIT operator SMSG commands issued above are:

```
GLD1190I Audit status
Option      Setting
AUDIT       ON
ERROR       MODIFY SEARCH
ALL         BIND
NONE
```

Working with Audit Records

The LDAP events are logged in an SMF file as type 83 subtype 3 records. The log record is a mixture of binary and EBCDIC data. The general format of SMF type 83 records is described in the chapter for SMF records, Record type 83: Security events in *z/VM: RACF Security Server Macros and Interfaces*.

You can use the RACF SMF Unload utility to reformat the LDAP SMF type 83 subtype 3 records for easier analysis. These audit records can be in the following different forms:

- A tabular format, suitable for import to a relational database manager.
- eXtensible Markup Language (XML) documents

Information on how to use the RACF SMF Unload utility can be found in *z/VM: RACF Security Server Auditor's Guide*. This document describes how the reformatted data can further be processed by SQL and sort/merge applications. Samples are also cited.

For the format and content of the LDAP SMF Audit records, see [Appendix B, “SMF records,”](#) on page 691.

Monitoring LDAP Server Resources

The LDAP server monitors the basic resources that it uses to ensure that they are still available. If a resource becomes unavailable, the LDAP server is configured to either terminate or to operate without the resource until the resource becomes available.

Server Backends During Startup

When the LDAP server is started, the server processes the LDAP server configuration file and then initializes each of the backends that are configured. If an error is detected during initialization of a backend, that backend is not usable. Based on the **srvStartUpError** option in the LDAP server configuration file, the LDAP server either shuts itself down or continues running with the those backends that successfully start. After the problem encountered during backend initialization is fixed, the LDAP server must be restarted to make that backend available. For the description of the **srvStartUpError** configuration option, see [“Step 6. Create and Customize the LDAP Configuration File \(DS CONF\)”](#) on page 83. The option does not apply to resource problems encountered after the LDAP server backends have started. The LDAP server response to those problems is described below.

Network Communications

The LDAP server uses TCP/IP for its client communications. The LDAP server also monitors TCP/IP and detects when one of the network interfaces in use by the server has failed. Based on the **tcpTerminate** option in the LDAP server configuration file, the LDAP server can then either shut itself down or try to reestablish the failed interface. For the description of the **tcpTerminate** configuration option, see [tcpTerminate](#).

The **tcpTerminate** option also controls the LDAP server response to a failure when initializing the SSL interfaces if it has been configured. The LDAP server either terminates or continues processing but does not use the failed interface. After the problem is fixed, the LDAP server must be restarted to make that interface available.

Client Connections

As the number of concurrent client connections approaches the maximum number of client connections allowed on the LDAP server, the LDAP server issues warning messages to the console when additional

clients attempt to bind to the LDAP server. To avoid overloading the console with messages, these warning messages are issued, at most, once per minute for a maximum of 60 times while the number of concurrent client connections remains at a high level. If the number of concurrent client connections on the LDAP server falls below a safe threshold, another console message is issued stating that the number of concurrent client connections is now at a safe level. After this, the cycle of warning messages can begin again if the number of concurrent client connections again approaches the maximum number of connections allowed on the LDAP server.

The issuance of these console warning messages on a fairly regular basis may signify that the **maxConnections** option in the LDAP server configuration file is set to a low value and should be increased. The activity log on the LDAP server can be used to monitor the number of client connections. For more information on activity logging, see [“Activity logging” on page 163](#).

File System

The LDAP server uses the Byte File System to store the directory information for the LDBM, GDBM, and CDBM backends. The LDAP server detects when file system errors occur, such as no space available or inability to write to required files or file directories. Based on the **fileTerminate** option in the LDAP server configuration file, the LDAP server can then either shut itself down or continue running with the affected LDBM, GDBM, or CDBM backend in read-only mode (updates to the directory are rejected). When the file system problem has been dealt with, the operator can use the LDAP server BACKEND operator SMSG command to change the LDBM, GDBM, or CDBM backend back to read-write mode. For the description of the **fileTerminate** configuration option, see [“Step 6. Create and Customize the LDAP Configuration File \(DS CONF\)” on page 83](#).

Running and Using the LDAP Backend Utilities

Utility programs are provided to assist in initializing and backing up the data managed by the LDAP server.

Operation	LDBM utility
Unload data from backend directory to an LDIF file	DS2LDIF
Encrypt passwords in a backend directory	DB2PWDEN
Perform an extended operation to configure and manage enhanced replication environments	LDAPEXOP

These programs can be run from CMS.

Format and usage information for the utilities are in:

- [“DS2LDIF \(ds2ldif utility\)” on page 175](#)
- [“DB2PWDEN \(db2pwdn utility\)” on page 172](#)
- [“LDAPEXOP \(ldapexop utility\)” on page 181](#).

Running the Backend Utilities in CMS

When started, DS2LDIF, DB2PWDEN, and LDAPEXOP read an environment variable file. The default file is DS ENVVARS. This default can be changed by setting the environment variable LDAP_DS_ENVVARS_FILE to the full path name of the desired environment variable file. Some of the environment variables that can be set are NLSPATH and LANG.

SSL/TLS Information for LDAP Utilities

The contents of a client's key database file is managed with the GSKKYMANT utility. For information about the GSKKYMANT utility, see [SSL Certificate Management in z/VM: TCP/IP User's Guide](#). The GSKKYMANT utility is used to define the set of trusted certification authorities (CAs) that are to be trusted by the client. By obtaining certificates from trusted CAs, storing them in the key database file, and marking them as

trusted, you can establish a trust relationship with LDAP servers that use certificates issued by one of the CAs that are marked as trusted.

If the LDAP servers accessed by the client use server authentication, it is sufficient to define one or more trusted root certificates in the key database file. With server authentication, the client can be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL/TLS connection with the server are encrypted, including the LDAP credentials that are supplied on the **ldap_sasl_bind_s()** API.

For example, if the LDAP server is using a high-assurance VeriSign certificate, you should obtain a CA certificate from VeriSign, receive it into your key database file, and mark it as trusted. If the LDAP server is using a self-signed GSKKMAN server certificate, the administrator of the LDAP server can supply you with a copy of the server's certificate request file. Receive the certificate request file into your key database file and mark it as trusted.

Using the DB2PWDEN or LDAPEXOP utilities without the **-Z** parameter and calling the secure port on an LDAP server (in other words, a nonsecure call to a secure port) is not supported. Also, a secure call to a nonsecure port is not supported.

SSL/TLS encrypts the keyring file. Either the password must be specified as part of the **-P** parameter or file specification of a stash file that is created using the GSKKMAN utility must be specified in the form `file://` followed immediately (no blanks in between) by the file specification of the stash file.

DB2PWDEN (db2pwdn utility)

Format

```
➤➤ DB2PWDEN — options ➤➤
```

Purpose

DB2PWDEN is provided to encrypt or hash all unencrypted, AES encrypted, and DES encrypted user passwords in an already loaded LDBM or CDBM backend. The utility runs as a client operation while the server is active, and causes the server to encrypt or hash all the `userPassword` attribute values that are unencrypted, AES encrypted, or DES encrypted with the `pwEncryption` method configured on the LDAP server. The utility must be run by an LDAP administrator with the appropriate authority or a user with the authority to update password values. See Administrative roles for more information about administrative role authority.

This utility must be run in a guest that has IPLed ZCMS.

Parameters

options

The following shows the *options* you can use for DB2PWDEN:

-?

Prints this text.

-b base

Uses *base* as the starting point for the update instead of the default. If **-b** is not specified, this utility examines the `LDAP_BASEDN` environment variable for a *base* definition.

Set the `LDAP_BASEDN` environment variable using Language Environment runtime environment variable `_CEE_ENVFILE`. For more information, see [z/OS: XL C/C++ Programming Guide](#).

-d debuglevel

Specifies the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server, as described in “Debug Levels” on page 107. [Table 21 on page 108](#) lists the specific debug levels. The default is no debug messages.

-D binddn

Uses *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a NULL string.

If the **-S** or **-m** option is equal to DIGEST-MD5 or CRAM-MD5, this option is the authorization DN which is used for making access checks. This directive is optional when used in this manner.

-g realmname

Specifies the *realmName* to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.

-h ldaphost

Specifies the host on which the LDAP server is running. The default is the local host.

-K keyfile

Specifies the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the **SSL_KEYRING** environment variable with an associated name.

SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, then the current directory is assumed to contain the file.

This parameter is ignored if **-Z** is not specified.

-m mechanism or -S mechanism

Specifies the bind method to use. You can use either **-m** or **-S** to indicate the bind method.

The default is SIMPLE. You can also specify EXTERNAL to indicate that a certificate (SASL external) bind is requested, CRAM-MD5 to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or DIGEST-MD5 to indicate a SASL digest bind is requested.

The EXTERNAL method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate is used.

The CRAM-MD5 method requires protocol level 3. The **-D** or **-U** option must be specified.

The DIGEST-MD5 method requires protocol level 3. The **-U** option must be specified. The **-D** option can optionally be used to specify the authorization DN.

-N keyfiledn

Specifies the label associated with the key in the key database file.

-p ldapport

Specifies the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.

-P keyfilepw

Specifies either the key database file password or the file specification for a SSL password stash file. When the stash file is used, it must be in the form *file://* followed immediately (no blanks) by the file system file specification (for example, *file:///etc/ldap/sslstashfile*). The stash file must be a BFS file.

This parameter is ignored if **-Z** is not specified.

-U username

Specifies the *userName* for CRAM-MD5 or DIGEST-MD5 binds. The *userName* is a short name (uid) that is used to perform bind authentication.

This option is required if the **-S** or **-m** option is set to DIGEST-MD5.

-w bindpasswd

Uses *bindpasswd* as the password for simple authentication. The default is a NULL string.

-v

Use verbose mode, with many diagnostics written to standard output.

-Z

Uses a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake.

The **-K** (*keyfile*) option or equivalent environment variable is required when the **-Z** option is specified. The **-P** (*keyfilepw*) option is required when the **-Z** option is specified and the key file specifies a file system key database file. If you choose to use a certificate that is different than the default specified in the key database, the **-N** (*keyfilelabel*) option must be specified.

All other command line inputs result in a syntax error message and the correct syntax is displayed. If the same option is specified more than once or if both **-m** and **-S** are specified, the last value specified is used.

The DB2PWDEN utility sends the **PasswordPolicy** control as a non-critical control when the user attempts to authenticate to the targeted LDAP server. If the bound user is subject to password policy on the LDAP server, the DB2PWDEN utility parses and displays the warning and error messages from the **PasswordPolicy** control response.

Examples

Following are some DB2PWDEN examples:

- The following command:

```
db2pwdn -D "cn=admin" -w secret
```

Encrypts all unencrypted, AES encrypted, or DES encrypted passwords in the LDBM or CDBM backend at the LDAP server on the local host. The base is defined in the LDAP_BASEDN environment variable. The encryption method used is the pwEncryption method configured on the LDAP server.

- The following command:

```
db2pwdn -h ushost -p 391 -D "cn=admin" -w secret
        -b "o=university, c=US"
```

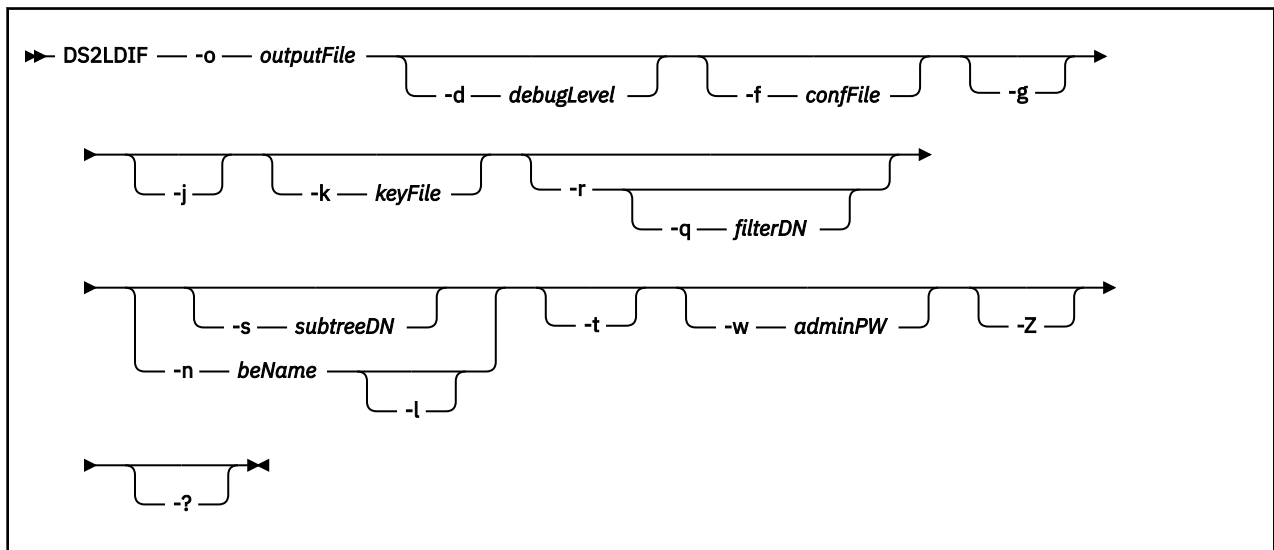
Encrypts all clear text user passwords starting at the base "o=university, c=US" in the LDBM backend on host ushost at port 391. The encryption method used is the pwEncryption method configured on the LDAP server.

Return codes

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being issued.

DS2LDIF (ds2ldif utility)

Format



Purpose

The DS2LDIF command is used to unload entries from a directory stored in an LDBM or CDBM backend into a file in LDAP Data Interchange Format (LDIF). The utility is also used to obtain the LDAP server schema entry. DS2LDIF cannot be used to unload a GDBM or SDBM backend.

Parameters

-d debugLevel

Specifies the level of the debug messages to be created. The *debugLevel* is specified in the same fashion as the debug level for the LDAP server, as described in [“Debug Levels” on page 107](#). The default is no debug messages.

-f confFile

Specifies the name of the LDAP server configuration file to use. This configuration file needs to contain only information for the backend to be unloaded. The DS CONF configuration file is used if the *-f* parameter is not specified.

-g

Specify to unload entries in genealogical order. This unloads entries in each subtree together, doing a depth-first traversal of the directory. Specify this option when you are unloading a large number of entries that you will later load using the **ldif2ds** utility with the *-g* option, because this order of entries improves the capacity of **ldif2ds** to process large numbers of entries. Unloading in this order requires more processing and impacts unload performance.

Note: The **ldif2ds** utility is not available with LDAP on z/VM, but the DS2LDIF output can be used on other platforms where **ldif2ds** is available; for example, on the IBM Tivoli Directory Server for z/OS.

-j

Indicates that the **replicateOperationalAttributes** control value is not written to the output LDIF file. The **replicateOperationalAttributes** control value has the **modifyTimestamp**, **createTimestamp**, **creatorsName**, and **modifiersName** attribute types and values for the entry base64 encoded.

-k keyFile

Specifies the name of the file containing the LDAP server encryption keys. The file specified by the LDAPKEYS FILEDEF statement is used if the *-k* option is not specified. The key file must be specified if any entries to be unloaded have **userPassword**, **secretKey**, **replicaCredentials**, **ibm-replicaKeyPw**, or **ibm-slappedMasterPw** attribute values that are encrypted using the DES or AES

algorithm. These attribute values are decrypted and base64 encoded as the entry is written to the output LDIF file.

-l

Specifies that entries under the `cn=localhost` suffix are unloaded from the LDBM backend. When an entire LDBM backend is unloaded, the entries under `cn=localhost` are not unloaded unless the **-l** option is specified. The **-l** option cannot be used when the **-s** option is specified.

-n beName

Specifies the name of the LDBM or CDBM backend to unload. *beName* is the name assigned to the backend on its **database** record in the LDAP server configuration file or the name that is automatically generated when the LDAP server is started. This can be used to indicate which LDBM or CDBM backend to process when there are multiple LDBM or CDBM backends in the configuration file. The **-n** option cannot be used when the **-s** option is specified.

-o outputFile

Specifies the fully-qualified output file to contain the unloaded directory entries.

-q filterDN

Specifies a distinguished name (DN) of a replication filter entry which contains **ibm-replicationFilterAttr** attribute values. These values are filters used to skip entire entries or attributes within entries while unloading the directory. For more information, see [Partial replication in z/VM: TCP/IP LDAP Administration Guide](#).

The targeted LDAP server must be running and the **-r** option must be specified when the **-q** option is specified. Also, the CDBM backend must be configured and **useAdvancedReplication on** specified in the CDBM backend section of the server configuration file to perform unload filtering.

-r

Perform an **unloadRequest** extended operation (1.3.18.0.2.12.62) to unload the subtree or backend. If the LDAP server that contains the backend that is to be unloaded is running, an **unloadRequest** extended operation can be sent to the LDAP server to unload the entries.

-s subtreeDN

Identifies the DN of the top entry of the subtree whose entries are to be unloaded. This entry, plus all below it in the directory hierarchy, are written to the output file. The **-s** option must be used to unload the LDAP server schema entry, `cn=schema`. The **-s** option cannot be used when the **-n** option is specified.

-t

Specifies that hashed **userPassword** and **ibm-slappedAdminPw** attribute values are unloaded with their encryption tag in clear text. See [Using the -t \(tagging\) option](#) for more information on this option.

-w adminPW

When using the **unloadRequest** extended operation, specify the password of the LDAP root administrator defined in the configuration file. Do not specify the **-w** option if the **adminPW** option is specified in the LDAP server configuration file. In this case, the value from the server configuration file is used to perform the LDAP bind before sending the **unloadRequest** extended operation to the LDAP server. If the **adminPW** configuration option is not present and the **-w** option is not specified or a **?** is specified on the **-w** option, a prompt is displayed for the LDAP root administrator password.

-Z

When using the **unloadRequest** extended operation, use SSL to encrypt the communication between DS2LDIF and the LDAP server. By default, DS2LDIF will attempt to use SSL to communicate with the LDAP server assuming that the LDAP server configuration file has the necessary SSL options (for example, `sslKeyRingFile`, `sslKeyRingFilePW`, `sslCertificate`, `sslKeyRingStashFile`) specified along with a secure listen option (for example, `listen ldaps://`). If SSL cannot be used, DS2LDIF fails.

-?

Displays command usage information.

All other command line inputs will result in a syntax error message, after which the proper syntax is displayed. Also, specifying the same option multiple times with different values will result in a syntax error.

Examples

The following example invokes DS2LDIF to unload all of the entries from the LDBM backend named `ldbm1` in the LDAP configuration file `DS.CONF`. The output is written to file `/ldbmdata/ldif.data`.

```
ds2ldif -o /ldbmdata/ldif.data -n ldbm1 -f //ds.conf
```

Using the -t (tagging) option: When the **-t** option is used on **ds2ldif** utility, the format of the unloaded **userPassword** or **ibm-slapdAdminPw** attribute depends on how the value is encrypted or hashed.

1. If the value is hashed using crypt, MD5, SHA, SHA224, SHA256, SHA384, or SHA512 one-way hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value is base64 encoded in the unloaded value. The format of the unloaded value is:

```
attrtype: {tag}base64encoded_and_hashedvalue
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**, *tag* is crypt, MD5, SHA, SHA224, SHA256, SHA384, or SHA512. For example:

```
userpassword: {crypt}0fCik9fUqZnixuKkYQ==
userpassword: {MD5}34d121/hie8s
userpassword: {SHA}24309gf[jgt
userpassword: {SHA224}1cf7ypKsUI0v2mK1ZKPQFPw7cSkUDjy5nqa/Eg==
userpassword: {SHA256}K7gNU3sdo+OL0wNhqoVWhr3g6s1xYv72o1/pe/Unols=
userpassword: {SHA384}WKd1ukESvjAFrkQHznV9iP2nHUBJe7gCbsrFTU4//HIyzo3jq1rLMK4
5dg/ufFPt
userpassword: {SHA512}vSsar3708Jvp9Szi2NWZZ02Bqp1qRCFpbCTZPdBhnWgs5WtNZKnvCXD
hztmeD2cmW192CF5bDufKRpayrW/isg==
```

2. If the value is hashed using SSHA (Salted SHA), SSHA224, SSHA256, SSHA384, or SSHA512 hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value and salt values are base64 encoded in the unloaded value. The format of the unloaded value is:

```
attrtype: {tag}base64encoded_hashed_and_salt_values
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**. *tag* is SSHA, SSHA224, SSHA256, SSHA384, or SHA512. For example:

```
userpassword: {SSHA}yEmjV/P10snkDbFMpXARCPuZA0evrN4xquMjGW5bMK1haAkb5Zt6VQ==
userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+X0cryro0otC3X3smmwN1Abs+
222PJRE1E7GJU24adtbpo=
userpassword: {SSHA256} qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA51AL3ePW2u01
Ur6q+Ye/UYJG+eOyaAuHEehFN30kGJwA==
userpassword: {SSHA384}mbP0pQkuXY1DswEDq6JYWP2Y95jgysAX0wohTmbKP74tQvnkR19G5e
u46qth1j0Kfvm7HItIuCzcdSRMTe80vynEsv+10eSfge60u3yrXs0cNeN/yw5yMp+FUX0HIg4f
userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdTkaRwhdBXDVep3Uj/WIRB+KB7zON8DX48gA
L1k1QCRnrLv0jvyBEB45Dmyj71AwT3M2T5PeagtoTixbDs1XgVH7zDqAHosWJEI0Zn0viQFP3Cx6
1R3OM0td5XEAJKC3RBTnhYkOXmdqqwe6KkorUdaMQ=
```

3. If the value is encrypted using a two-way encryption algorithm (DES or AES) or is not encrypted, then the unencrypted value is base64 encoded in the unloaded value and there is no tag. The format of the unloaded value is:

```
attrtype:: base64encoded_and_unencryptedvalue
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**. For example:

```
userpassword:: kfa6903axs
```

This is also the format used when unloading **secretKey**, **replicaCredentials**, **ibm-replicaKeyPwD**, and **ibm-slapdMasterPw** attribute values, because these values can only be encrypted using two-way encryption algorithms.

Note:

1. The LDAP server loads and uses tagged **userPassword** or **ibm-slapdAdminPw** values that are hashed in crypt, SHA, SSHA (Salted SHA), or MD5 and were unloaded using **ds2ldif** utility with the **-t** option. Also, these tagged values might be acceptable for other LDAP providers to load into their directory. If

it is not directly loadable, this format is easily modifiable for loading by another provider into its LDAP directory.

2. The values returned by the **crypt** algorithm are portable only to other X/Open-conformant systems when the **pwCryptCompat** configuration option is set to **off**. When the **pwCryptCompat** configuration option is set to **off**, the **crypt()** algorithm uses ASCII, which is a subset of UTF-8, when generating the hashed **userPassword** or **ibm-slapdAdminPw** attribute values. Therefore, it is recommended that the **pwCryptCompat** configuration option be set to **off** when it is necessary to share **userPassword** or **ibm-slapdAdminPw** attribute values hashed in **crypt()** between the z/VM LDAP server and other ASCII-based LDAP servers. For more information on the **pwCryptCompat** configuration option, see [pwCryptCompat](#).

When the **-t** option is not used on **ds2ldif** utility, the format of the unloaded password attributes depends upon how the **userPassword** or **ibm-slapdAdminPw** value is encrypted or hashed.

1. If the value is hashed using crypt, SHA, or MD5 one-way hashing algorithms, then the tag and the hashed **userPassword** or **ibm-slapdAdminPw** values are base64 encoded in the unloaded value. The format of the unloaded value is:

```
attrtype:: base64encodedValue
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**. *base64encodedValue* is a base64 encoding of

```
{tag}encryptedvalue
```

where, *tag* is **crypt**, **MD5**, or **SHA**. For example:

```
userpassword:: e2NyeXB0fdHwopPX1KmZ4sbipGE=
```

2. If the value is hashed using SHA224, SHA256, SHA384, or SHA512 one-way hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value is base64 encoded in the unloaded value. The format of the unloaded value is:

```
attrtype: {tag} base64encoded_and_hashedvalue
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**. *tag* is **SHA224**, **SSHA256**, **SSHA384**, or **SHA512**. For example:

```
userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+X0cryro0otC3X3smmwN1Abs+
222PJRE1E7GJU24adtbpo=
userpassword: {SSHA256}qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA51AL3ePW2u01
Ur6q+Ye/UYJG+e0yaAuHEehFN30kGJwA==
userpassword: {SSHA384}mbP0pQkuXY1DswEDq6JYWp2Y95jgysAX0wohTmbKP74tQvnkR19G5e
u46qth1j0Kfvm7HIItIuCzcdSRMTe80vynEsv+10eSfge60u3yrXs0cNeN/yw5yMp+FUX0HIg4f
userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdTkaRwhdBXDVEP3Uj/WIRB+KB7zON8DX48gA
L1k1QCRnrlv0jvyBEB45Dmyj71Awrt3M2T5PeagtoTixbDs1XgVH7zDqAHosWJEI0Zn0viQFP3Cx6
1R3OM0td5XEAJKC3RBThYk0Xmdqqwe6KkorUdaMQ=
```

3. If the value is hashed using SSHA (Salted SHA), SSHA224, SSHA256, SSHA384, or SSHA512 one-way hashing algorithm, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value is base64 encoded in the unloaded value. The format of the unloaded value is:

```
attrtype: {SSHA}base64encoded_hashed_and_salt_values
```

where, *attrtype* is **userpassword** or **ibm-slapdadminpw**. *tag* is **SSHA**, **SSHA224**, **SSHA256**, **SSHA384**, or **SSHA512**. For example:

```
userpassword: {SSHA}yEmjV/P10snkDbFMpXARCPuzA0evrN4xquMjGW5bMK1haAkb5Zt6VQ==
userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+X0cryro0otC3X3smmwN1Abs+
222PJRE1E7GJU24adtbpo=
userpassword: {SSHA256}qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA51AL3ePW2u01
Ur6q+Ye/UYJG+e0yaAuHEehFN30kGJwA==
userpassword: {SSHA384}mbP0pQkuXY1DswEDq6JYWp2Y95jgysAX0wohTmbKP74tQvnkR19G5e
u46qth1j0Kfvm7HIItIuCzcdSRMTe80vynEsv+10eSfge60u3yrXs0cNeN/yw5yMp+FUX0HIg4f
userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdTkaRwhdBXDVEP3Uj/WIRB+KB7zON8DX48gA
```

```
L1k1QCRnrLv0jvyBEB45Dmyj71AwT3M2T5PeagtoTIXbDs1XgVH7zDqAHosWJEI0Zn0viQFP3Cx6
1R30M0td5XEAJKC3RBTnhYk0Xmdqqwe6KkoιUdaMQ=
```

4. If the value is encrypted using a two-way encryption algorithm (DES or AES) or is not encrypted, then the unencrypted value is base64 encoded in the unloaded value and there is no tag present. The format of the unloaded value is:

```
attrtype:: base64encoded_and_unencryptedvalue
```

where, *attrtype* is **userpassword** or **ibm-slappedadminpw**. For example:

```
userpassword:: e01ENXkfa6903axs
```

This is also the format used when unloading **secretKey**, **replicaCredentials**, **ibm-replicaKeyPwd**, and **ibm-slappedMasterPw** attribute values, because these values can only be encrypted using two-way encryption algorithms.

Using the unloadRequest extended operation: The **unloadRequest** extended operation is used to remotely unload directory data from a currently running z/VM or z/OS LDAP server. The **unloadRequest** extended operation is required when attempting to unload data from an LDBM or CDBM backend that is already running with an active z/VM or z/OS LDAP server. The **unloadRequest** extended operation is also required when using the **-q filterDN** option to filter entries as they are being unloaded from an LDBM or CDBM backend. The **-x** option can be used to force the **unloadRequest** extended operation to be sent to the z/VM or z/OS LDAP server for any unload operation.

The **ds2ldif** utility does the following when an **unloadRequest** extended operation is performed:

1. An LDAP root administrator simple bind is attempted using each secure **listen** option in the LDAP server configuration file until a successful secure connection is established with the LDAP server. The **sslKeyRingFile** option in the LDAP server configuration file indicates the key database file that DS2LDIF uses to communicate securely with the LDAP server. The **sslKeyRingFilePW** or **sslKeyRingPWStashFile** options in the LDAP server configuration file are used by DS2LDIF to gain access to the key database file. The **sslCertificate** option in the LDAP server configuration file is used as the SSL certificate when DS2LDIF establishes the secure connection with the LDAP server. The **adminDN** that is specified in the LDAP server configuration file is used as the bind DN. If there is an **adminPW** configuration option present, it is used as the password. If there is no **adminPW** configuration option, DS2LDIF uses the value of the DS2LDIF **-w** option or issues a prompt for the password if the **-w** option is not specified.

Note:

- a. The **ds2ldif** utility sends the **PasswordPolicy** as a non-critical control when the LDAP root administrator attempts to authenticate to the LDAP server. If the LDAP root administrator's entry is subject to password policy on the targeted LDAP server, the DS2LDIF utility parses and displays the warning and error messages from the **PasswordPolicy** control response.
 - b. The routine used to prompt for the password returns at most 255 characters, truncating any additional characters. If the length of the administrator password is greater than 255, you must either have an **adminPW** configuration option or specify the **-w** option.
 - c. Ensure that the userid running **ds2ldif** utility has access to the key database file that is specified on the **sslKeyRingFile** option.
2. If a secure connection is not established with the LDAP server or there are no secure listen options in the LDAP server configuration file, an LDAP root administrator simple bind is attempted using each nonsecure listen option until a successful nonsecure connection is established. If a connection is not established with the LDAP server, DS2LDIF ends.

Note:

- a. In order to perform the **unloadRequest** extended operation, it is required that the bound user be the LDAP root administrator.
- b. If the **-Z** option is specified, DS2LDIF communicates over a secure port if a connection is established. If a secure connection is not established, DS2LDIF fails.

3. Entries are unloaded into an output file on the LDAP server's system. The name of the file is specified in the DS2LDIF **-o** option.

Note: If **DS2LDIF** unloads entries when using the **unloadRequest** extended operation, the output file is produced in the code page that is specified in the LANG environment variable of the LDAP server. Otherwise, the output file is produced in the code page that is specified in the LANG environment variable of **DS2LDIF**.

4. An **unloadResponse** extended operation is sent back to DS2LDIF indicating how many entries are unloaded and whether the extended operation is successful.

Additional setup when Unloading LDBM or CDBM directory data: If DS2LDIF is being used to unload LDBM or CDBM directory data and an **unloadRequest** extended operation is not being performed (the LDAP server is not running), the user ID that is running DS2LDIF must have read and execute access to the directory that is specified on the **databaseDirectory** configuration option of the LDBM or CDBM backend that is being unloaded and it must also have read access to the LDBM or CDBM database and checkpoint files in that directory to successfully perform an unload. Also, the user ID must be in the owning GID group or be the owning UID for the directory and files in that directory to successfully perform an unload.

If **DS2LDIF** is being used to unload directory data and an **unloadRequest** extended operation is not being performed (the LDAP server is not running), the user ID that is running **DS2LDIF** must have read and execute access to the directory specified by the **schemaPath** configuration option or its default value and it must also have read access to the schema.db file in the directory. Also, the user ID must be in the owning GID group or be the owning UID for the directory and schema.db file in that directory to successfully perform an unload.

Usage

1. If DS2LDIF is invoked with neither the **-s** nor the **-n** option and there is only one LDBM or CDBM backend in the LDAP server configuration file, all of the directory entries (other than **cn=localhost** unless the **-l** option is specified) in that particular LDBM or CDBM backend are unloaded.
2. DS2LDIF only unloads owner and ACL information for entries that have a specific owner or ACL. Any entry data with an inherited owner or ACL will not have owner or ACL information unloaded.
3. The **ibm-entryuuid** attribute is included in each entry unloaded by DS2LDIF. The LDAPADD command can be used to load an entry containing the **ibm-entryuuid** attribute only if the LDAP server is running in maintenance mode and the LDAPADD is performed by the LDAP root administrator or replica administrator (when using replication). If you do not need the **ibm-entryuuid** attribute in the loaded entry to have the same value as in the entry that was unloaded, then remove the attribute from each entry in the DS2LDIF output file and a new **ibm-entryuuid** value will be assigned when the entry is loaded using the LDAPADD command.
4. If there are password policy operational attribute values present in the entries that are being unloaded, they are included in the LDIF file created by the DS2LDIF utility. The password policy operational attributes are **pwdChangedTime**, **pwdAccountLockedTime**, **pwdExpirationWarned**, **pwdFailureTime**, **pwdGraceUseTime**, **pwdHistory**, **pwdReset**, **ibm-pwdAccountLocked**, **ibm-pwdIndividualPolicyDN**, and **ibm-pwdGroupPolicyDN**.
5. If password policy is active in the LDAP server and there are plans to populate a new LDAP server with the same data, you might want the password policies in the **cn=ibmpolicies** suffix in the CDBM backend be unloaded along with the wanted data in the LDBM backend. These unloaded password policy entries must be added to the new server under the **cn=ibmpolicies** suffix in case there are user or group entries that have **ibm-pwdgrouppolicydn** or **ibm-pwdindividualpolicydn** attribute values. These user or group entries cannot be added to the server with the LDAPADD utility until all referenced password policy entries are added to the CDBM backend.
6. LDAP 3 has a related set of Internet Drafts which documents the introduction of a version mechanism for use in creating LDIF files. The **ds2ldif** utility always creates "tagged" LDIF files. The LDIF tag consists of a single line at the top of the LDIF file:

```
version: 1
```


All characters contained in the version one LDIF file are portable characters represented in the local code page that is specified in the LANG environment variable of the LDAP server. If you reload the data in the LDIF file, ensure the utility that loads the data expects the data in the same code page you used to create the LDIF file. Strings containing nonportable characters (for example, textual values containing multi-byte UTF-8 characters) are base64 encoded.

7. To unload the LDAP server schema entry, specify:

```
-s cn=schema
```

The schema entry is unloaded in LDIF modify format. The current LDAP server schema entry is unloaded.

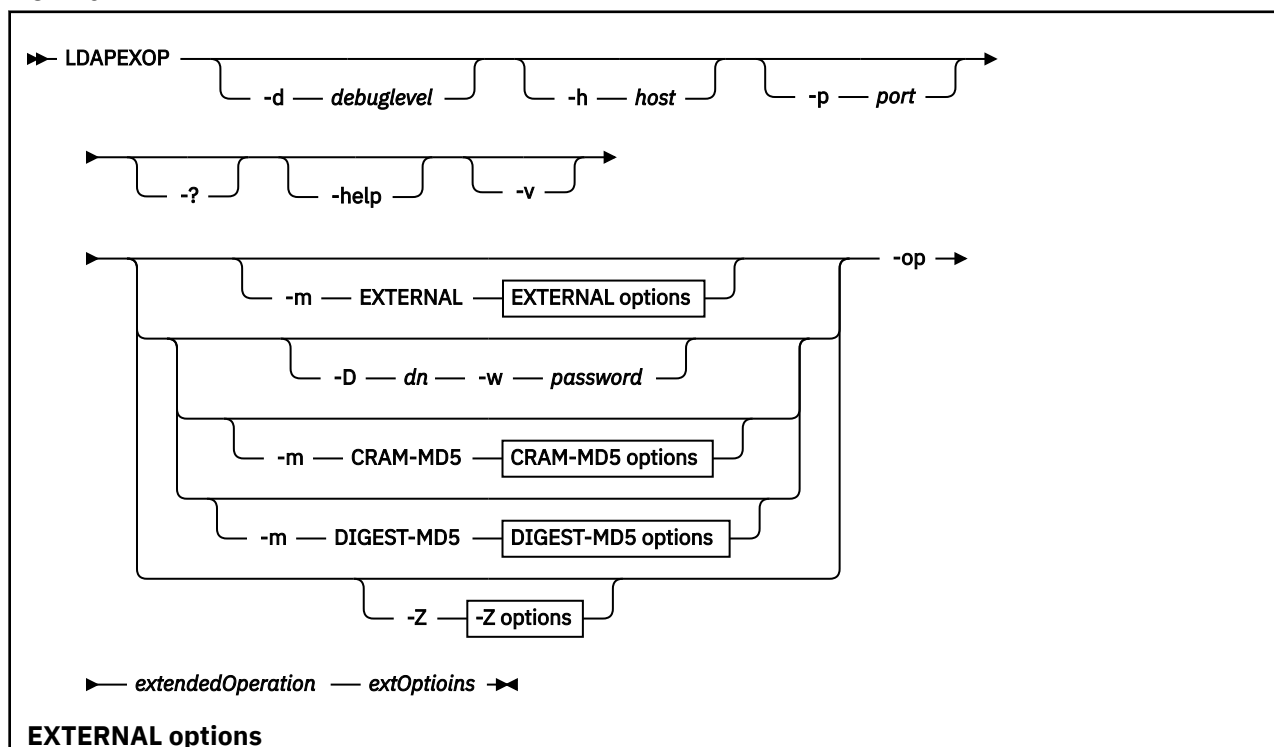
8. When you edit the output file produced by DS2LDIF, you should use an editor that does not delete blanks at the end of a line. If the output file contains a line that ends with blanks, using such an editor will result in deleting the blanks, therefore, changing the value of the attribute. This is very important when the line is continued, since the existing blanks will no longer separate the last word in the line from the continuation on the next line. The maximum line length in a DS2LDIF output file is 77; continued lines are always 77 long when the file is created by DS2LDIF.
9. The **LDAP_DEBUG** environment variable can also be used to set the debug level for DS2LDIF. For more information on specifying the debug level, see [“Debug Levels”](#) on page 107.
10. If you are using DS2LDIF to unload a large number of entries that you plan to later load using **ldif2ds**, specify the **-g** option on both DS2LDIF and **ldif2ds**. This causes DS2LDIF to unload the entries using a depth-first traversal of the directory; the root is unloaded, then each subtree directly below the root is traversed. This order results in improved **ldif2ds** load capacity, at the cost of some impact on DS2LDIF performance. LDAPADD performance does not benefit from the **-g** option for DS2LDIF.

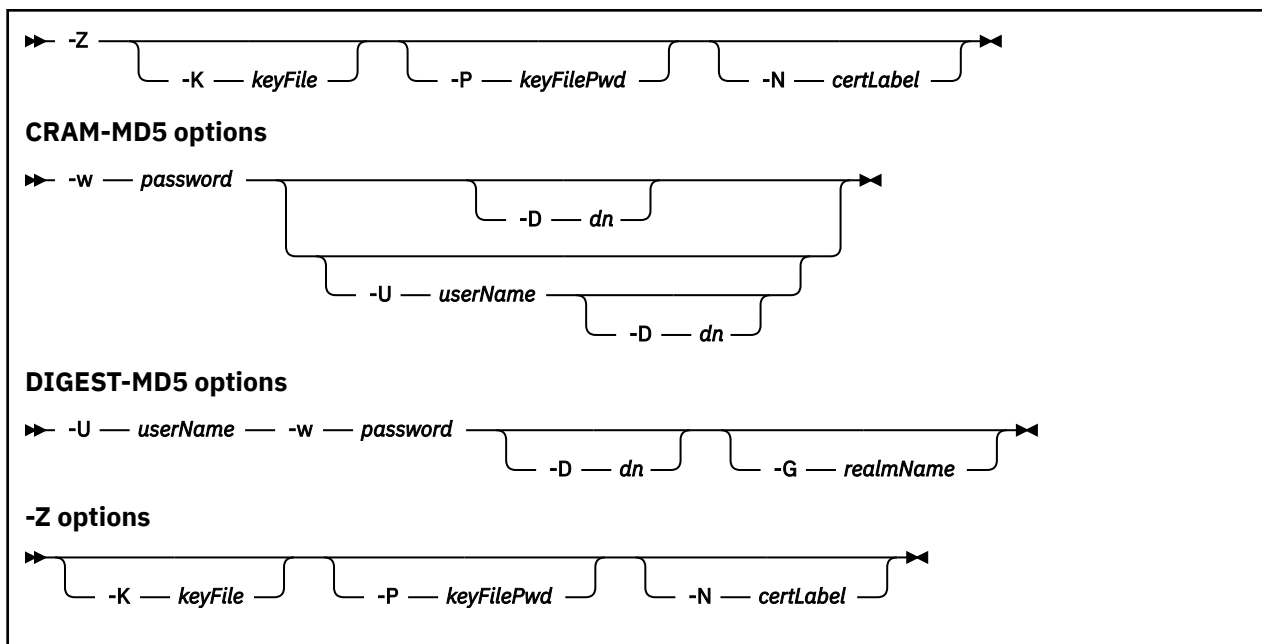
Return codes

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being issued.

LDAPEXOP (ldapexop utility)

Format





Purpose

The LDAPEXOP utility provides an interface to the ldap extended operation() API. LDAPEXOP runs as a client operation while the server is active. This utility must be run in a guest that has IPLed ZCMS.

The LDAPEXOP utility performs the following extended operations:

- **acctstatus - Account status** extended operation
- **cascrepl - Cascading control replication** extended operation
- **controlqueue - Control replication queue** extended operation
- **controlrepl - Control replication** extended operation
- **controlreplerr - Control replication error log** extended operation
- **effectpwdpolicy - Effective password policy** extended operation
- **geteffectiveacl - GetEffectiveACL** extended operation
- **getusertype - User type** extended operation
- **quiesce - Quiesce or unquiesce context** extended operation
- **repltopology - Replication topology** extended operation

Parameters

The options for the LDAPEXOP utility are divided into two categories:

1. There are general options that specify how to connect to the LDAP server. These options must be specified before the `-op` option. For an explanation about the general options, see [General options](#).
2. There are extended operation options that identify the extended operation to be performed. These options are specified after the general options and must begin with the `-op` option. For an explanation about these options, see [Extended operations options](#).

Option names can normally be specified in a reduced manner with as few as one or two of the initial characters specified.

All other command-line inputs result in a syntax error message, after the correct syntax is displayed. Except for the `-op` option, if the same option is specified multiple times, the last value specified is used. The `-op` option must be specified only once.

General options: These options must be specified before the `-op` option.

-?

Display the usage.

-d

Specify the level of debug messages to be created. The *debugLevel* is specified in the same fashion as the debug level for the LDAP server. See [“Debug Levels” on page 107](#). The default is no debug messages.

-D

Specify the DN to use to bind to the LDAP directory. The *dn* parameter is a string-represented DN. The default is a NULL string.

If the *-m* option is equal to DIGEST-MD5 or CRAM-MD5, this option is the authorization DN which is used for making access checks. This directive is optional when used in this manner.

-G

Specify the realm name to use when doing a DIGEST-MD5 bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a DIGEST-MD5 challenge; otherwise, it is optional.

-h

Specify the hostname or IP address on which the LDAP server is running. The default is the local host.

-help

Display the usage.

-K

Specify the name of the SSL key database file. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name.

If *keyFile* is not a fully-qualified file name, the current directory is assumed to contain the key database file.

This parameter is ignored if *-Z* is not specified.

-m

Specify the bind method to use.

Specify EXTERNAL to indicate that a certificate (SASL external) bind is requested, CRAM-MD5 to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or DIGEST-MD5 to indicate a SASL digest bind is requested.

The EXTERNAL method requires a protocol level of 3. You must also specify *-Z*, *-K*, and *-P* to use certificate bind. Unless you want to use the default certificate in the key database file, use the *-N* option to specify the label of the certificate.

The CRAM-MD5 method requires protocol level 3. The *-D* or *-U* option must be specified.

The DIGEST-MD5 method requires protocol level 3. The *-U* option must be specified. The *-D* option can optionally be used to specify the authorization DN.

If *-m* is not specified, a simple bind is performed.

-N

Specify the label associated with the certificate in the key database file.

This parameter is ignored if *-Z* is not specified.

-p

Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.

-P

Specify either the key database file password or the file specification for an SSL password stash file. When the stash file is used, it must be in the form `file://` followed immediately (no blanks) by the file system file specification (for example, `file:///etc/ldap/sslstashfile`). The stash file must be a BFS file.

This parameter is ignored if -Z is not specified.

-U

Specify the user name for CRAM-MD5 or DIGEST-MD5 binds. The *userName* is a short name (uid) that is used to perform bind authentication.

This option is required if the -m option is set to DIGEST-MD5.

-v

Use verbose mode, with many diagnostics written to standard output.

-w

Specify the bind password for simple, CRAM-MD5, and DIGEST-MD5 authentication. The default is a NULL string.

Specify ? to prompt for the password.

-Z

Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake.

The -K *keyFile* option or equivalent environment variable is required when the -Z option is specified. The -P *keyFilePwd* option is required when the -Z option is specified and the key file specifies a file system key database file. Unless you want to use the default certificate in the key database file, use the -N option to specify the label of the certificate.

Extended operations options: The -op *extendedOperation* option identifies the extended operation to be performed. The *extOptions* indicate the required and optional options for the specific extended operation. Each extended operation that is supported by the LDAPEXOP utility is documented in detail in ["Supported extended operations"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

The supported extended operations and their options are:

acctstatus -d *userDn*

Perform the **Account status** extended operation. This extended operation returns the status of a user entry with a **userPassword** attribute value. The status returned is either opened, locked, or expired. For more information about the **Account status** extended operation, see ["Account status"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

-d *userDn*

Specifies the DN of the entry to be queried.

This example performs the **Account status** extended operation to query the status of the cn=jon,o=acme,c=us entry.

```
ldapexop -D adminDn -w adminPw -op acctstatus
-d cn=jon,o=acme,c=us
```

cascrepl -action { quiesce | unquiesce | replnow | wait } -rc *contextDn* [-timeout *secs*]

Perform the **Cascading control replication** extended operation. The requested action is applied to the specified server and also passed along to all replicas of the given context (subtree). If any of these are forwarding replicas or gateway servers, they pass the extended operation along to their replicas. The operation cascades over the entire advanced replication topology. For more information about the **Cascading control replication** extended operation, see ["Cascading control replication"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

-action { quiesce | unquiesce | replnow | wait }

Specifies the **Cascading control replication** extended operation action to be performed.

quiesce

Halts further updates, except by replication. Client updates under the specified context are restricted to the LDAP administrator, if using the **Server Administration** control (OID 1.3.18.0.2.10.15), and any replication master DN's with authority under this context.

unquiesce

Resumes normal operation, client updates are accepted.

replnow

Replicates all queued changes to all replica servers as soon as possible, regardless of schedule. This operation is propagated to the consumer server of each replication agreement without waiting for all queued updates to be applied.

wait

Replicates all queued changes to all replica servers as soon as possible, regardless of schedule. This operation is propagated to the consumer server of each replication agreement after all queued updates for that agreement are applied.

-rc contextDn

Specifies the DN of the advanced replication context (subtree).

-timeout secs

Specifies the number of seconds that the extended operation has to successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

The following example performs the **Cascading control replication** extended operation to wait for all updates to be replicated to each consumer server in the `o=acme, c=us` replication context. If the extended operation does not successfully complete in 60 seconds, the operation ends with a return code of `LDAP_TIMEOUT`.

```
ldapexop -D adminDn -w adminPw -op cascrepl -action wait
-rc "o=acme,c=us" -timeout 60
```

controlqueue -skip {all | changeId} -ra agreementDn

Perform the **Control replication queue** extended operation. This extended operation indicates which pending changes in the advanced replication queue are skipped and not replicated to the consumer server. For more information about the **Control replication queue** extended operation, see ["Control replication queue"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

-skip {all | changeId}**all**

Deletes (skips) all changes currently queued for replication for the specified replication agreement DN.

changeId

Deletes (skips) the change queued for replication with the change ID matching the specified number. The number must be 1 - 4294967295. Change IDs can be determined by searching the *agreementDn* entry for the **ibm-replicationPendingChanges** operational attribute. Only the next change to be replicated can be skipped in this manner.

-ra agreementDn

Specifies the DN of the advanced replication agreement.

The following example performs the **Control replication queue** extended operation to delete (skip) all pending updates in the advanced replication queue for the replication agreement, `cn=server3, ibm-replicaSubentry=master1-id, ibm-replicaGroup=default, o=acme, c=us`. This extended operation prevents all changes in the replication queue from being replicated to all consumer servers.

```
ldapexop -D adminDn -w adminPw -op controlqueue -skip all
-ra "cn=server3,ibm-replicaSubentry=master1-id,
ibm-replicaGroup=default,o=acme,c=us"
```

controlrepl -action {suspend | resume | replnow} { -rc contextDn | -ra agreementDn }

Perform the **Control replication** extended operation. This extended operation indicates whether to suspend, resume, or immediately replicate entries in the advanced replication context (subtree) or agreement. For more information about the **Control replication** extended operation, see ["Control replication"](#) in *z/VM: TCP/IP LDAP Administration Guide*.

-action { suspend | resume | replnow }

Specifies the **Control replication** extended operation action to be performed.

suspend

Suspends advanced replication for the replication agreement or context (subtree). Any updates under the replication agreement or context are queued until the **Control replication** extended operation is performed to resume updates to consumer servers.

resume

Resumes advanced replication for the replication agreement or context (subtree).

replnow

Replicates immediately any outstanding updates for the replication agreement or context (subtree), regardless of schedule. `replnow` has no effect on a suspended replication agreement or context.

-rc contextDn | -ra agreementDn

Specifies the DN of the advanced replication context (subtree) or agreement the action is to be performed against. To perform the action against all agreements under a replication context, use `-rc contextDn` with the DN of the replication context. Alternatively, to perform this action against a single agreement, use `-ra agreementDn` with the DN of the replication agreement. A *contextDn* and an *agreementDn* cannot both be specified.

The following example performs the **Control replication** extended operation to suspend advanced replication on the replication agreement, `cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us`.

```
ldapexop -D adminDn -w adminPw -op controlrepl
-action suspend
-ra "cn=server3,ibm-replicaSubentry=master1-id,
ibm-replicaGroup=default,o=acme,c=us"
```

controlreplerr -ra agreementDn { [-delete {failureId | all}] | [-retry {failureId | all}] | [-show failureId] }

Perform the **Control replication error log** extended operation. This extended operation indicates whether to delete, retry, or show a failed advanced replication update. Failing operations can be determined by searching the *agreementDn* entry for the **ibm-replicationFailedChanges** operational attribute which contains the failing change ID. For more information about the **Control replication error log** extended operation, see ["Control replication error log" in z/VM: TCP/IP LDAP Administration Guide](#).

-ra agreementDn

Specifies the DN of the advanced replication agreement.

-delete { failureId | all }

Removes one or all failed replication updates.

failureId

Deletes only the failed update specified by the *failureId* for this agreement. The *failureId* must be 1 - 4294967295.

all

Deletes all the failed updates for this agreement.

-retry { failureId | all }

Reprocesses one or all failed replication updates.

failureId

Retries only the failed update specified by the *failureId* for this agreement. The *failureId* must be 1 - 4294967295.

all

Retries all the failed updates for this agreement.

-show failureId

Shows the failed update specified by the *failureId* for this agreement. The *failureId* must be 1 - 4294967295.

The following example performs the **Control replication error log** extended operation to remove all failures from the replication error log for the replication agreement, `cn=server3,ibm-replicaSubentry=master1-id, ibm-replicaGroup=default,o=acme,c=us`.

```
ldapexop -D adminDn -w adminPw -op controlreplerr
-delete all
-ra "cn=server3,ibm-replicaSubentry=master1-id,
ibm-replicaGroup=default,o=acme,c=us"
```

effectpwdpolicy -d entryDn

Perform the **Effective password policy** extended operation. This extended operation returns the effective password policy entries and the attribute values of the effective password policy that the specified entry is subject to. For more information about the **Effective password policy** extended operation, see ["Effective password policy" in z/VM: TCP/IP LDAP Administration Guide](#).

-d entryDn

Specifies the DN of the entry to be queried to obtain its effective password policy.

The following example performs the **Effective password policy** extended operation to query the effective password policy of the `cn=jon,o=acme,c=us` entry.

```
ldapexop -D adminDn -w adminPw -op effectpwdpolicy
-d cn=jon,o=acme,c=us
```

geteffectiveacl

Perform the **GetEffectiveACL** extended operation. This extended operation performs a search using the base DN, scope, and search filter to retrieve a list of entries. For each entry in the list, the extended operation then uses the optional bind and entry access information to determine the effective access control and returns the following:

- The entry DN to which access was requested
- The subject and all of its alternate DNs and group DNs for which access was calculated
- The source attribute values (**aclEntry**, **aclPropagate**, **aclSource**, **entryOwner**, **ownerPropagate**, and **ownerSource**) in effect for the entry
- The applicable attribute values (**aclEntry** and **entryOwner**) used to form the effective permissions
- The calculated effective access class permissions
- The calculated effective attribute permissions.

Note: Administrator permissions can be restricted by ACL filters that match the administrator DN. As a result, the calculated effective permissions of the administrator DN can have an effect on the results returned from the extended operation.

```
geteffectiveacl -filt searchFilter
[-b baseDN]
[-s {base | one | sub}]
[-a {never | always | search | find}]
[-z sizeLimit]
[-l timeLimit]
[-m {SIMPLE | CRAM-MD5 | DIGEST-MD5 | EXTERNAL}]
[-dn DN]
[-u racfUserID]
[-ip bindIP]
[-time accessTime]
[-day accessDay]
[-en]
```

Referral entries are not returned from the search operation. Therefore, access to referral entries is not calculated.

-filt[er] searchFilter

An IETF RFC 2254-compliant LDAP search filter that determines the entries for which to calculate effective permissions (*The String Representation of LDAP Search Filters*). The `searchFilter` is required.

-b[ase] baseDN

The base DN used as a starting point for calculating effective permissions. If unspecified, *baseDN* is set to the empty string "", representing a null-based subtree starting point.

-s[cope] scope

The scope and *baseDN* determine the directory entries to which DN's permissions are calculated for, starting with *baseDN*. The scope must be **base**, **one**, or **sub**. Scope is required only if *baseDN* is unspecified. Otherwise, the default is **sub**.

-a deref

Specifies how alias dereferencing is done during the search operation. *deref* must be one of **never**, **always**, **search**, or **find** to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is **never**.

-z sizeLimit

Indicates the maximum number of entries to calculate effective permissions. A value of 0 indicates that there is no limit. The LDAP server can also provide a size limit on the number of entries returned in the **sizeLimit** configuration option. For more information about the LDAP server size limit, see [sizeLimit](#). If this option is not specified or is set to 0, the maximum number of entries returned is limited only by the LDAP server limit.

-l timeLimit

Indicates the maximum wait time (in seconds) for the search. A value of 0 indicates that there is no time limit on the search request. The LDAP server can also provide a limit on the search time in the **timeLimit** configuration option. For more information about the LDAP server time limit, see [page timeLimit](#). If this option is not specified or is set to 0, the maximum number of seconds is limited only by the LDAP server limit.

-m[echanism] {SIMPLE | EXTERNAL | CRAM-MD5 | DIGEST-MD5}

The string representation of the mechanism the DN used to bind to the LDAP server. This is used to match the **ibm-filterBindMechanism** attribute in any ACL filters. The default value is **SIMPLE**.

-dn DN

Indicates the string representation of the DN for which to calculate effective permissions. This is the bind DN for **SIMPLE**, **CRAM-MD5**, or **DIGEST-MD5** bind, and the subject DN for **EXTERNAL** bind. If **SIMPLE** bind is specified and this option is unspecified, the effective permissions are calculated for an anonymous bind. If **CRAM-MD5**, **DIGEST-MD5**, or **EXTERNAL** is used, this option is required.

-u[serid] racfUserID

The string representation for the RACF user ID associated with the subject DN in the SSL certificate. This value is optional when **EXTERNAL** is specified. For all other bind mechanisms, this option is ignored.

-ip bindIP

The string representation of the IPv4 or IPv6 address of the client used to connect to the LDAP server. If this value is unspecified, it defaults to the unspecified IPv4 address, 0.0.0.0.

-time accessTimeOfDay

The string representation of the time of day the directory entry is accessed. The format is *hh:mm*, where *hh* ranges from 00 to 23, and *mm* ranges from 00 to 59. If this value is unspecified, it defaults to the server's current time of day.

-day accessDayOfWeek

The string representation of the day of week the directory entry is accessed. Valid values are 0 - 6, where: *Sunday* = 0, *Monday* = 1, *Tuesday* = 2, *Wednesday* = 3, *Thursday* = 4, *Friday* = 5, *Saturday* = 6. If this value is unspecified, it defaults to the server's current day of week.

-en[ryption]

When specified, it assumes the bind DN uses a secure LDAP connection (SSL). Otherwise, it assumes the bind DN does not use a secure LDAP connection (SSL) with a non-clear cipher specification.

The following example performs the **GetEffectiveACL** extended operation for each entry returned on the subtree search of the `dc=yourcompany, dc=com` subtree. The requested subtree search uses `dc=yourcompany, dc=com` as the *baseDN*, with a filter of `"objectclass=*"`, a search size limit of 100, a search time limit of 10 seconds, and no alias dereferencing. Based on these returned search entries, the **GetEffectiveACL** extended operation calculates the effective ACLs for user `cn=Joe Shmoe, ou=users, dc=yourcompany, dc=com` when a simple bind is done from IP address `129.176.132.92` at `18:30` on a Saturday over a secure SSL connection. For the output of this example, see ["Querying effective permissions" in *z/VM: TCP/IP LDAP Administration Guide*](#).

```
ldapexop -D adminDn -w adminPw -op geteffectiveacl
-filter "objectclass=*" -base "dc=yourcompany,dc=com" -s sub -a never
-z 100 -l 10 -dn "cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com"
-ip 129.176.132.92 -time 18:30 -day 6 -mech SIMPLE -encrypt
```

getusertype

Perform the **User type** extended operation. This extended operation provides the authenticated user with their user type and administrative roles. See ["Supported extended operations" in *z/VM: TCP/IP LDAP Administration Guide*](#).

This example performs the **User type** extended operation to determine the user type and roles for user `cn=jon,o=ibm,c=us`:

```
ldapexop -D cn=jon,o=ibm,c=us -w secret -op getusertype
```

quiesce -rc contextDn [-end]

Perform the **Quiesce or unquiesce context** extended operation. When an advanced replication context is quiesced, update operations are accepted only from certain users. For more information about the **Quiesce or unquiesce context** extended operation, see ["Quiesce or unquiesce context" in *z/VM: TCP/IP LDAP Administration Guide*](#).

-rc contextDn

Specifies the DN of the advanced replication context (subtree) to be quiesced or unquiesced.

-end

Unquiesces the advanced replication context (subtree). If not specified, the default is to quiesce the replication context.

The following example performs the **Quiesce or unquiesce context** extended operation to quiesce the replication context `o=acme, c=us`. After this extended operation is performed, updates to entries within the replication context are only allowed from certain users.

```
ldapexop -D adminDn -w adminPw -op quiesce
-rc "o=acme,c=us"
```

repltopology -rc contextDn [-timeout secs] [-ra agreementDn]

Perform the **Replication topology** extended operation. This extended operation synchronizes all replication topology related entries under the specified context DN. For more information about the **Replication topology** extended operation, see ["Replication topology" in *z/VM: TCP/IP LDAP Administration Guide*](#).

-rc contextDn

Specifies the DN of the advanced replication context (subtree) on the supplier server for which advanced replication topology related entries are synchronized. The extended operation is cascaded through all forwarding and gateway servers.

-timeout secs

Specifies the number of seconds that the extended operation has to successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

-ra agreementDn

Specifies the DN of the advanced replication agreement and allows the operation to be restricted to only one server and its consumer.

The following example performs the **Replication topology** extended operation to synchronize replication topology entries within the `o=acme, c=us` replication context with the consumer

defined in the replication agreement `cn=server3,ibm-replicaSubentry=master1-id, ibm-replicaGroup=default,o=acme,c=us`. This extended operation does not cascade, it only modifies the consumer defined in the replication agreement before returning.

```
ldapexop -D adminDn -w adminPw -op repltopology -rc  
"o=acme,c=us"  
-ra "cn=server3,ibm-replicaSubentry=master1-id,  
ibm-replicaGroup=default,o=acme,c=us"
```

Usage

1. You must be bound as an LDAP administrator with the appropriate authority to successfully perform all extended operations (other than the User type extended operation) against the z/VM LDAP server. See ["Administrative group and roles" in z/VM: TCP/IP LDAP Administration Guide](#) for information about which extended operations are allowed to be performed by administrative group members.
2. The LDAPEXOP utility sends the **PasswordPolicy** control as a non-critical control when the user attempts to authenticate to the targeted LDAP server. If the bound user is subject to password policy on the LDAP server, the LDAPEXOP utility parses and displays the warning and error messages from the **PasswordPolicy** control response.
3. The LDAP_DEBUG environment variable can be used to set the debug level. For more information about specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see ["Debug Levels" on page 107](#).
4. You can specify an LDAP URL for host on the **-h** parameter. For more information, see ["LDAP URLs and LDAP Filters" in z/VM: TCP/IP User's Guide](#).

Return codes

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being issued.

For additional diagnostic information about each of the extended operations, see ["Supported extended operations" in z/VM: TCP/IP LDAP Administration Guide](#).

Internationalization Support

This topic discusses translated messages and UTF-8 support.

Translated Messages

The **LANG** and **NLSPATH** environment variables are set for the LDAP server and utility programs in the LDAP environment variables file, DS ENVVARS.

A sample DS ENVVARS file is shipped on the TCPMAINT 591 disk as LDAPDS SAMPENVR. You can copy this file to TCPMAINT 198 as DS ENVVARS and modify its contents. Following is part of the sample file:

```
NLSPATH=/usr/lib/nls/msg/%L/%N  
LANG=En_US.IBM-1047
```

There are no default values for these variables. Messages are also available in Japanese. The **LANG** variable should be set to `LANG=Ja_JP.IBM-939`. These variables can also be set in the CENV group of GLOBALV.

UTF-8 Support

UTF stands for "UCS (Unicode) Transformation Format". The UTF-8 encoding can be used to represent any Unicode character. Depending on a Unicode character's numeric value, the corresponding UTF-8 character is a 1, 2, or 3 byte sequence. [Table 27 on page 191](#) shows the mapping between Unicode and UTF-8. For more information on UTF-8, see IETF RFC 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*.

Table 27. Mapping between Unicode and UTF-8

Unicode range (hexadecimal)	UTF-8 octet sequence (binary)
0000-007F	0xxxxxxx
0080-07FF	110xxxxx 10xxxxxx
0800-FFFF	1110xxxx 10xxxxxx 10xxxxxx

LDAP 3 specifies that all data exchanged between LDAP clients and servers be UTF-8. The LDAP server supports UTF-8 data exchange as part of its LDAP 3 support.

Chapter 8. Configuring the MPRoute Server

This chapter describes the Multiple Protocol Routing (MPRoute) server. Before presenting the details of the configuration process, some specifics on the dynamic routing protocols supported by MPRoute are discussed. This information will help you decide if this application is suitable for your network.

Much of the configuration information must be the same on all routers that communicate with one another, so it is critical that the server administrator work closely with the network administration team to ensure that the proper protocol is used and that the correct protocol settings are entered.

Understanding MPRoute

MPRoute supports the RIP and OSPF routing protocols for both IPv4 and IPv6. You can send either RIP Version 1 or RIP Version 2 packets, but not both at the same time, on a single interface. However, you can configure a RIP interface to receive packets for both versions.

MPRoute has the following characteristics:

- MPRoute's job is limited to the management of the TCP/IP stack routing table. MPRoute is not directly involved in the actual routing decisions made by the TCP/IP stack when routing a packet to its destination.
- A one-to-one relationship exists between an instance of MPRoute and a TCP/IP stack.
- If IPv4 interfaces are defined in the MPRoute configuration file as OSPF or RIP interfaces, all IPv4 dynamic routes are deleted from the stack's IPv4 routing table upon initialization of MPRoute. If IPv6 interfaces are defined in the MPRoute configuration file as OSPF or RIP interfaces, all IPv6 dynamic routes, excluding those dynamic routes learned through IPv6 router discovery, are deleted from the stack's IPv6 routing table upon initialization of MPRoute. MPRoute then repopulates the stack routing tables that it cleared, using information learned through the routing protocols.
- MPRoute allows the generation of multiple, equal-cost routes to a destination, thus providing load-splitting support. Up to sixteen equal-cost routes to a destination are allowed.
- IPv4 Internet Control Message Protocol (ICMP) redirects are ignored when MPRoute is active and there are IPv4 interfaces configured to MPRoute as RIP or OSPF interfaces. IPv6 ICMP redirects are ignored when MPRoute is active and there are IPv6 interfaces configured to MPRoute as RIP or OSPF interfaces.
- The IPv4 Maximum Transmission Unit (MTU) and subnet mask and destination address must be specified using the `OSPF_INTERFACE`, `RIP_INTERFACE`, and `INTERFACE` statements in the MPRoute configuration file. Note, however, that if the MTU has been configured on the `LINK` statement in the TCP/IP configuration file, MPRoute uses that value rather than anything configured in the MPRoute configuration file for that interface.
- MPRoute uses the virtual machine console for its logging and tracing.
- If a given dynamic routing protocol is to be used on a particular interface, then the corresponding configuration statement must exist in MPRoute's configuration file.
- If you are not configuring all of your interfaces through the `INTERFACE` statement, and the defaults for the `INTERFACE` statement are not acceptable for interfaces (such as VIPA interfaces) that are not involved in the communication of RIP or OSPF protocols to MPRoute, then you need to specify `GLOBAL_OPTIONS IGNORE_UNDEFINED_INTERFACES=YES`. For instance, the defaults for `INTERFACE` are never going to be acceptable for point-to-point interfaces because the MPRoute server has no other way of knowing the `DESTINATION_ADDR`. So, you should use `GLOBAL_OPTIONS IGNORE_UNDEFINED_INTERFACES=YES` for point-to-point interfaces.
- MPRoute supports Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The virtual routes are included in the advertisements to adjacent routers. Adjacent routers learn about virtual routes from the advertisements and can use them to reach the destinations on the z/VM host.

- MPRoute is supported only in a virtual machine running a single processor. Attempting to run the MPRoute virtual machine with multiple virtual processors may result in performance degradation.
- z/VM supports RFC 4191 as a type B host, which defines a Default Router Preference field for received Router advertisement messages, and indicates whether the sending router should be preferred (or not) over other default routers. It also supports the Default Router Preference field when acting as a router and sending router advertisement messages.
- z/VM supports the Expanded Flags Option for the incoming Router Advertisement messages (defined in RFC 5175), which expands the available number of flag bits by adding an additional 48 flag bits.
- TCP/IP will monitor the responsiveness of MPRoute and will terminate the MPRoute server if it becomes unresponsive. The server will be restarted if it is in the AUTOLOG list.

The use of static routes (defined through the GATEWAY TCP/IP configuration statement) with MPRoute is not recommended. Static routes might interfere with the discovery of better routes to a destination or, if the destination should become unreachable through the static route, inhibit the ability to switch to another route. If, however, you must define static routes, all static routes are considered to be of equal cost and static routes are not replaced by OSPF or RIP routes. Use extreme care when working with static routes and MPRoute. If you want static routes to be advertised to other routers, set `IMPORT_STATIC_ROUTES` to YES on the `AS_BOUNDARY_ROUTING` and `IPV6_AS_BOUNDARY_ROUTING` configuration statements or set `SEND_STATIC_ROUTES` to YES on the `RIP_INTERFACE` and `IPV6_RIP_INTERFACE` configuration statements.

For details on the statements in the MPRoute configuration file, see [“MPRoute configuration file”](#) on page 203.

You can perform MPRoute server administration tasks through the SMSG interface. For details, see [“SMSG Interface to the MPRoute Server”](#) on page 261.

Dynamic routing

For dynamic routing, z/VM uses a server running MPRoute, a multiprotocol router. For IPv4, MPRoute supports the RIP Version 1, RIP Version 2, and OSPF routing protocols. You can send RIP Version 1 or RIP Version 2, but not both at the same time, on a single interface. However, you can configure a RIP interface to receive both versions.

For IPv6, MPRoute supports the IPv6 OSPF and IPv6 RIP protocols.

Note: If you previously used the Routed server to provide dynamic routing services for your installation, you need to migrate to the MPRoute server for these services.

IPv4 dynamic routing using MPRoute

For IPv4, MPRoute implements the OSPF protocol described in RFC 1583 (*OSPF Version 2*), and the RIP protocols described in RFC 1058 (*Routing Information Protocol*) and in RFC 1723 (*RIP Version 2 - Carrying Additional Information*). It provides an alternative to the static TCP/IP gateway definitions. The host running with MPRoute becomes an active OSPF or RIP router in a TCP/IP network. Either or both of these routing protocols can be used to dynamically maintain the host routing table. For example, MPRoute can detect when a route is created, is temporarily unavailable, or if a more efficient route exists. If both OSPF and RIP protocols are used simultaneously, OSPF routes will be preferred over RIP routes to the same destination.

Open Shortest Path First (OSPF)

OSPF is classified as an Interior Gateway Protocol (IGP). This means that it distributes routing information between routers belonging to a single autonomous system (AS), a group of routers all using a common routing protocol. The OSPF protocol is based on link-state or shortest path first (SPF) technology. It has been designed expressly for the TCP/IP Internet environment, including explicit support for IP subnetting and the tagging of externally derived routing information.

OSPF performs the following tasks:

Multiple routes

Provides support for up to 16 equal-cost routes.

Authentication

Provides for the authentication of routing updates.

IP multicast

Uses IP multicast when sending or receiving the updates.

Allows network grouping

Allows sets of networks to be grouped together. Such a grouping is called an area. The topology of an area is hidden from the rest of the autonomous system. This method of hiding information enables a significant reduction in routing traffic. Also, routing within the area is determined only by the area's own topology, lending the area protection from bad routing data. An area is a generalization of an IP subnetted network.

IP subnet configuration

Enables the flexible configuration of IP subnets. Each route distributed by OSPF has a destination and mask. Two different subnets of the same IP network number may have different sizes (that is, different masks). This is commonly referred to as variable length subnetting. A packet is routed to the best (longest or most specific) match. Host routes are considered to be subnets whose masks are all ones (0xFFFFFFFF).

Authenticate OSPF protocol exchanges

Can be configured such that all OSPF protocol exchanges are authenticated. This means that only trusted routers can participate in the autonomous system's routing. A single authentication scheme is configured for each physical link. This enables some links to use authentication while others do not.

OSPF is a dynamic routing protocol. It quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of convergence. This period of convergence is short and involves a minimum of routing traffic as compared to the RIP protocol.

In a link-state routing protocol, each router maintains a database describing the autonomous system's topology. Each individual piece of this database is a particular router's local state (for example, the router's usable interfaces and reachable neighbors). The router distributes its local state throughout the autonomous system by flooding.

To generate routes, all routers run the exact same algorithm, in parallel. From the topological database, each router constructs a tree of shortest paths with itself as root. This shortest-path tree gives the route to each destination in the autonomous system. Externally derived routing information (for example, routes learned from the RIP protocol) appears on the tree as leaves. When several equal-cost routes to a destination exist, the routes (up to 16) are added to the TCP/IP stack's route table.

Externally derived routing data (for example, routes learned from the RIP protocol) is passed transparently throughout the autonomous system. This externally derived data is kept separate from the OSPF protocol's link state data. Each external route can also be tagged by the advertising router, but not by MPRoute, enabling the passing of additional information between routers on the boundaries of the autonomous system. MPRoute does pass tags created by others. For information on configuring OSPF, see [“Configuration Steps for the MPRoute Server” on page 200](#).

Routing Information Protocol (RIP)

RIP is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. RIP is based on the Bellman-Ford or the distance-vector algorithm. RIP has many limitations and is not suitable for every TCP/IP environment. Before using the RIP function in MPRoute, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. For more information about how you can obtain information on RFCs, see [Appendix D, “Related Protocol Specifications,” on page 715](#).

RIP uses the number of hops, or hop count, to determine the best possible route to a host or network. The term hop count is also referred to as the metric. In RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by RIP to 15 gateways.

A RIP router broadcasts routing information to its directly connected networks every 30 seconds. It receives updates from neighboring RIP routers every 30 seconds and uses the information contained in these updates to maintain the routing table. If an update has not been received from a neighboring RIP router in 180 seconds, a RIP router assumes that the neighboring RIP router is down, sets all routes through that router to a metric of 16 (infinity), and stops using those routes when routing IP packets. If an update has still not been received from the neighboring RIP router after another 120 seconds, the RIP router deletes from the routing table all of the routes through that neighboring RIP router.

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags

The route tags are used to separate *internal* RIP routes (routes for networks within the RIP routing domain) from *external* RIP routes, which may have been imported from an EGP (external gateway protocol) or another IGP. MPRoute does not generate route tags, but preserves them in received routes and readvertises them when necessary.

Variable subnetting support

Variable length subnet masks are included in routing information so that dynamically added routes to destinations outside subnetworks or networks can be reached.

Immediate next hop for shorter paths

Next hop IP addresses, whenever applicable, are included in the routing information to eliminate packets being routed through extra hops in the network.

Multicasting to reduce load on hosts

IP multicast address 224.0.0.9, reserved for RIP Version 2 packets, is used to reduce unnecessary load on hosts which are not listening for RIP Version 2 messages. This support is dependent on interfaces that are multicast-capable.

Authentication for routing update security

Authentication keys can be configured for inclusion in outgoing RIP Version 2 packets. Incoming RIP Version 2 packets are checked against the configured keys.

Configuration switches for RIP Version 1 and RIP Version 2 packets

Configuration parameters allow for controlling which version of RIP packets are to be sent or received over each interface.

Supernetting support

The supernetting feature is part of Classless InterDomain Routing (CIDR). Supernetting provides a way to combine multiple network routes into fewer supernet routes, thus reducing the number of routes in the routing table and in advertisements.

For configuration information for RIP, see [“Configuration Steps for the MPRoute Server” on page 200](#).

IPv6 dynamic routing using MPRoute

For IPv6, MPRoute implements the IPv6 RIP protocol described in RFC 2080 (*RIPng for IPv6*) and the IPv6 OSPF protocol described in RFC 2740 (*OSPF for IPv6*). It provides an alternative to the static TCP/IP gateway definitions. The host running with MPRoute becomes an active OSPF or RIP router in a TCP/IP network. Either or both of these routing protocols can be used to dynamically maintain the host IPv6 routing table. For example, MPRoute can detect when a route is created, is temporarily unavailable, or if a more efficient route exists. If both IPv6 OSPF and IPv6 RIP protocols are used simultaneously, IPv6 OSPF routes will be preferred over IPv6 RIP routes to the same destination.

IPv6 OSPF protocol

IPv6 OSPF is classified as an Interior Gateway Protocol (IGP). This means that it distributes routing information between routers belonging to a single autonomous system (AS), a group of routers all using a common routing protocol. The IPv6 OSPF protocol is based on link-state or shortest path first (SPF) technology.

IPv6 OSPF is a dynamic routing protocol. It quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of convergence. This period of

convergence is short and involves a minimum of routing traffic as compared to the IPv6 RIP protocol. However, it does generally require more CPU resources on participating routers.

In IPv6 OSPF, sets of networks can be placed together in groups called areas. The topology of an area is hidden from the rest of the AS. This method of hiding information enables a significant reduction in routing traffic. Also, routing within the area is determined only by the area's own topology, lending the area protection from bad routing data.

Each IPv6 OSPF router maintains a database describing the autonomous system's topology. Each individual piece of this database is a particular router's local state (for example, the router's usable interfaces and reachable neighbors). The router distributes its local state information by flooding. The routing information in an area is summarized and advertised into adjacent areas, allowing for the generation of interarea routes. This summarization and advertising of routing information between areas is the responsibility of the routers that reside on the border between areas (called area border routers).

To generate routes, all routers run the exact same algorithm, in parallel. From the topological database, each router constructs a tree of shortest paths with itself as root. This shortest-path tree gives the route to each destination in the autonomous system. Externally derived routing information (for example, routes learned from the IPv6 RIP protocol) appears on the tree as leaves. When several equal-cost routes to a destination exist, the routes (up to 16) are added to the TCP/IP stack's route table.

Externally derived routing data is passed transparently throughout the autonomous system. This externally derived data is kept separate from the IPv6 OSPF protocol's link state data. Each external route can also be tagged by the advertising router, but not by MPRoute, enabling the passing of additional information between routers on the boundaries of the autonomous system. MPRoute does pass tags created by others. For information on configuring IPv6 OSPF, see [“Configuration Steps for the MPRoute Server”](#) on page 200.

IPv6 RIP protocol

IPv6 RIP is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. IPv6 RIP is based on the Bellman-Ford or the distance-vector algorithm. IPv6 RIP has limitations and is not suited for every TCP/IP environment. Before using the IPv6 RIP function in MPRoute, read RFC 2080 to decide if RIP can be used to manage the IPv6 routing tables of your network. For more information about RFC 2080, see [Appendix D, “Related Protocol Specifications,”](#) on page 715.

IPv6 RIP uses the number of hops, or hop count, to determine the best possible route to a host or network. The term hop count is also referred to as the metric. In IPv6 RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by IPv6 RIP to 15 gateways.

A IPv6 RIP router broadcasts routing information to its directly connected networks every 30 seconds. It receives updates from neighboring IPv6 RIP routers every 30 seconds and uses the information contained in these updates to maintain the IPv6 routing table. If an IPv6 RIP update has not been received from a neighboring router in 180 seconds, an IPv6 RIP router assumes that the neighboring router is down, sets all IPv6 RIP routes through that router to a metric of 16 (infinity), and stops using those routes when routing IP packets. If an update has still not been received from the neighboring router after another 120 seconds, the IPv6 RIP router deletes from the IPv6 routing table all of the IPv6 RIP routes through that neighboring router.

Next hop IP addresses, whenever applicable, are included in IPv6 RIP updates to eliminate packets being routed through extra hops in the network. IPv6 multicast address FF02::9, reserved for IPv6 RIP packets, is used to reduce unnecessary load on hosts that are not listening for IPv6 RIP messages.

For configuration information for IPv6 RIP, see [“Configuration Steps for the MPRoute Server”](#) on page 200.

Using RIP, IPv6 RIP, OSPF, and IPv6 OSPF with MPRoute

When MPRoute is initialized, it uses the MPRoute configuration file to determine which routing protocols will be enabled. If at least one OSPF interface is configured, the OSPF protocol is enabled. If at least one RIP interface is configured, RIP is enabled. If at least one IPv6 RIP interface is configured, IPv6 RIP is

enabled. If at least one IPv6 OSPF interface is configured, IPv6 OSPF is enabled. If MPRoute is started with no interfaces defined for a particular protocol, that protocol is disabled until one of the following occurs:

- MPRoute is stopped and restarted with a configuration file containing at least one interface of the specific type.
- MPRoute is dynamically reconfigured using the SMSG command with a configuration file containing at least one interface of the specific type.

When MPRoute is configured for both the OSPF and RIP protocols (either IPv4 or IPv6), routes that are learned through the OSPF protocol take precedence over routes learned through the RIP protocol.

The OSPF and RIP protocols are communicated over IPv4 interfaces that are defined with the `OSPF_INTERFACE` and `RIP_INTERFACE` configuration statements, respectively. An IPv4 interface involved in the communication of neither the RIP nor the OSPF protocol should be configured to MPRoute with the `INTERFACE` configuration statement, unless `Ignore_Undefined_Interfaces=YES` is coded on the `Global_Options` configuration statement. MPRoute supports a maximum of 255 real, physical, IPv4 interfaces (that is, interfaces on which data can actually be sent and received). There is no theoretical limit on how many VIPAs can be configured, though there are practical limits imposed by network design. For special VIPA considerations, see [“Notes on defining IPv4 interfaces”](#) on page 207.

The IPv6 OSPF and IPv6 RIP protocols are communicated over IPv6 interfaces that are defined with the `IPv6_OSPF_INTERFACE` and `IPv6_RIP_INTERFACE` configuration statements, respectively. An IPv6 interface not involved in the communication of the IPv6 OSPF or IPv6 RIP protocol can be configured to MPRoute with the `IPv6_INTERFACE` configuration statement. If default values are acceptable and you do not need to define additional prefixes to an IPv6 interface not involved in the communication of the IPv6 OSPF or IPv6 RIP protocol, it is not necessary to configure the interface to MPRoute at all. MPRoute will learn about the interface and its MTU value from the stack and use default values for other parameters. This is different from IPv4, where all interfaces should be configured to prevent MPRoute from using default values for MTU size and subnet mask, unless `Global_Options` is coded with `Ignore_Undefined_Interfaces=YES`.

MPRoute allows for the generation of multiple, equal-cost routes to a destination. For OSPF and IPv6 OSPF, up to 16 multiple equal-cost routes are allowed. For RIP and IPv6 RIP, multiple equal-cost routes are supported only to directly connected destinations over redundant interfaces.

Preventing futile neighbor state loops during adjacency formation

In OSPF environments in which there can be a problem with some remote hardware (for example, a router, switch, or network cable) that is beyond detection by z/VM hardware or software, MPRoute can get into an infinite (or futile) neighbor state loop over one of its interfaces with a neighbor. This loop can contribute to increased workload. In LAN configurations in which there are parallel OSPF interfaces that can reach the same neighbor for adjacency formation, unless you are using MPRoute futile neighbor state loop detection or unless you manually fix the problem, the backup interfaces are not used until after an outage occurs for the OSPF interface that was initially involved in an adjacency formation attempt with a designated router. For more information, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Futile neighbor state loops can occur during adjacency formation with a neighboring designated router before full adjacency has been reached. According to OSPF protocol, all routers on a multiaccess network attempt to establish adjacency with the designated router (DR) and backup designated router (BDR) after two-way communication has been established. When a problem occurs during the adjacency formation process, the adjacency formation attempt is restarted. Repeated adjacency attempts can continue until full adjacency has been established or an interruption occurs. Interruption occurs from an interface outage to the neighbor or from a neighbor outage. Whether or not parallel interfaces are used, MPRoute repeatedly attempts to establish full adjacency with the neighbor, and if the problem is due to a cabling error or failure or some other non-transient condition, it is highly unlikely that the problem will be resolved and the loop will become futile.

To stop a futile neighbor state loop, you can either manually deactivate the interface or suspend the OSPF interface within MPRoute so that MPRoute stops attempting to form the adjacency over the interface. If the OSPF interface is suspended, any active sessions using static routes over the interface are not

disrupted. If the interface is deactivated, all active sessions over the interface are disrupted. If a parallel OSPF interface is available, MPRoute then attempts to form adjacency with the neighbor over the parallel interface.

MPRoute futile neighbor state loop detection removes the need for manual detection of futile neighbor state loops, manual intervention to resolve the loops, and the disruption of existing sessions due to deactivating the interface. Code the `DR_MAX_ADJ_ATTEMPT` parameter on the OSPF or IPV6_OSPF statement, or both, in your MPRoute configuration file to enable this function. MPRoute will then report and control futile neighbor state loops during the adjacency formation process. Futile neighbor state loop detection function is supported on OSD, HIPERS, and LCS type devices.

If you use the `DR_MAX_ADJ_ATTEMPT` parameter, futile neighbor state loops are automatically detected and reported using message EZZ8157I. If a parallel OSPF interface is not available, adjacency formation attempts continue to be retried over the same interface. If parallel OSPF interfaces are available, an interface change is reported using message EZZ8158I and adjacency formation attempts are retried over a parallel OSPF interface. The problematic interface is suspended within MPRoute, but the interface is not deactivated and active sessions over the interface are not disrupted.

After a problematic OSPF interface is suspended in MPRoute and adjacencies are formed on a parallel OSPF interface, you might want to switch back to the original interface once you have fixed the problem that caused the futile neighbor state loop. To accomplish this, activate the repaired interface and then suspend the parallel interface. Once OSPF adjacencies are established over the repaired interface, the parallel interface can be reactivated so it is once again available as a backup for the repaired interface. Use the `ACTIVATE` option of the `SMSG MPRROUTE` command to activate a suspended OSPF interface. Use the `SUSPEND` option of the `SMSG MPRROUTE` command to manually suspend an OSPF interface. For more information, see [“SMSG Interface to the MPRoute Server” on page 261](#).

Special considerations

This topic discusses special considerations for virtual IP addresses and multiple equal-cost routes.

Virtual IP Addresses (VIPA)

MPRoute supports Virtual IP Addressing (VIPA) for IPv4 to handle network interface failures by switching to alternate paths. The VIPA routes are included in the IPv4 OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about VIPA routes from the advertisements and can use them to reach the destinations at the z/VM host.

Multiple equal-cost routes

When MPRoute is being used to provide dynamic routing for a TCP/IP stack, multiple routes to the same destination can be dynamically added to the TCP/IP stack's route table, based upon the routing information learned from other routers. These multiple routes will be added when the route calculation for each has resulted in the same route cost value. No more than 16 equal-cost routes will be added for each destination. For RIP and IPv6 RIP, multiple equal-cost routes will be added only to directly-connected destinations over redundant interfaces. The RIP and IPv6 RIP protocols will generate no more than one indirect route to a destination.

<i>Table 28. Multipath route limitations</i>			
Multipath route type	GATEWAY (Static)	MPRoute (OSPF and IPv6 OSPF)	MPRoute (RIP and IPv6 RIP)
Direct host	Yes (no limit)	Yes (up to 16)	No
Indirect host	Yes (no limit)	Yes (up to 16)	No
Direct network	Yes (no limit)	Yes (up to 16)	Yes (up to 16 for redundant interfaces)
Indirect network	Yes (no limit)	Yes (up to 16)	No

Table 28. Multipath route limitations (continued)

Multipath route type	GATEWAY (Static)	MPRoute (OSPF and IPv6 OSPF)	MPRoute (RIP and IPv6 RIP)
Default (indirect)	Yes (no limit)	Yes (up to 16)	No

Note: If you want more than 16 equal-cost routes for OSPF or if you want multiple equal-cost indirect routes for RIP, use the GATEWAY statement. Likewise, if you want more than 16 equal-cost routes for IPv6 OSPF or if you want multiple equal-cost indirect routes for IPv6 RIP, use the GATEWAY statement.

Dynamic Server Operation

The MPRoute server provides a VM Special Message (MSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information see [“MSG Interface to the MPRoute Server”](#) on page 261.

Configuration Steps for the MPRoute Server

MPRoute Server Configuration Steps
1. Update the TCP/IP server configuration file 2. Update the ETC SERVICES file. 3. Create the MPRoute configuration file (MPROUTE CONFIG). 4. Optional: Update the DTCPARMS file for the MPRoute server 5. Optional: Create static routes 6. Optional, if using the IPv4 OSPF protocol: Configure OSPF authentication.

Step 1. Update the TCP/IP server configuration file

Follow these substeps:

1. Include the name of the MPRoute server in the OBEY statement because of its dependence on the Raw Sockets.
2. Include the MPRoute server virtual machine user ID in the AUTOLOG statement of the TCP/IP server configuration file. The MPRoute server is then automatically started when TCP/IP is initialized. The IBM default user ID for this server is MPRROUTE.

Example: If you use the IBM default user ID, code:

```
AUTOLOG
MPROUTE 0
```

3. If you use the RIP protocol in MPRoute, reserve UDP port 520. Verify that the following statement is in your TCP/IP server configuration file:

```
PORT
520 UDP MPRROUTE ; MPRoute Server
```

4. If you use the IPv6 RIP protocol in MPRoute, reserve UDP port 521. Verify that the following statement is in your TCP/IP server configuration file:

```
PORT
521 UDP MPRROUTE ; MPRoute Server
```

5. When coding the HOME statement for IPv4 devices, ensure that each link has only one home address, and ensure that the home address is unique.

Autolog Considerations for MPRROUTE

If a server in the AUTOLOG list also has a PORT Statement reserving a TCP or UDP port but does not have a listening connection on that port, TCP/IP will periodically attempt to cancel and then restart that server. Therefore, if MPRoute is being started with AUTOLOG and only the OSPF protocol is being used (no RIP or IPv6 RIP protocols, and therefore no listening connection on the RIP and IPv6 RIP UDP ports), you **must** do one of the following:

- Ensure that the RIP UDP port (520) and the IPv6 RIP UDP port (521) are not reserved by the PORT statement in the TCP/IP server configuration file.
- Add the NOAUTOLOG parameter to the PORT statement in the TCP/IP server configuration file. For example,

```
PORT
520 UDP MPRROUTE NOAUTOLOG
```

or

```
PORT
521 UDP MPRROUTE NOAUTOLOG
```

Note: When using only the OSPF protocol, the autostart feature of AUTOLOG can be used as described above. TCP/IP will monitor the responsiveness of MPRoute and will terminate the MPRoute server if it becomes unresponsive. MPRoute will be restarted if it is in the AUTOLOG list.

If you fail to take one of the above actions, MPRoute will be periodically cancelled and restarted by TCP/IP.

Step 2. Update the ETC SERVICES file

The ETC SERVICES file contains the relationship between services and port numbers.

Follow these substeps:

1. If the RIP protocol of MPRoute is going to be used, ensure that the following line is added to the ETC SERVICES file:

```
route      520/udp      router MPRROUTE
route      521/udp      ipv6rip
```

Note: In the above example, the default well-known ports (520 and 521) are used.

2. Compare the port number configured in the ETC SERVICES file to the port number configured for the MPRoute server in the TCP/IP server configuration file to ensure that the port numbers are the same.

Step 3. Create the MPRoute Configuration File

The MPRoute configuration file contains the configuration statements through which you configure OSPF and RIP. A sample of this configuration file is shipped as MPRROUTE SCONFIG on TCPMAINT 591. Copy MPRROUTE SCONFIG to TCPMAINT 198, customize the file, and rename it to MPRROUTE CONFIG.

Like all configuration files, the system uses the first MPRROUTE CONFIG file found in the CMS search order.

Note: If you wish to have MPRROUTE use something other than the default name of MPRROUTE CONFIG for its configuration file, you can use the :Config. tag in the appropriate entry of the DTCPARMS file. For more information about this tag, refer to [Table 7 on page 37](#).

To create the MPRoute configuration file and find syntax information for OSPF and RIP configuration statements, see [“MPRoute configuration file” on page 203](#).

Step 4. Optional: Update the DTCPARMS File

When the MPRoute server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server.

Follow these substeps:

1. Check or code the following tags that affect the MRoute server:

- :NICK.
- :PARMS.
- :CONFIG.
- :TIMEZONE.

For more information on these tags, see [Table 7 on page 37](#).

2. If you need to do more customization than what is available in the DTCPARMS file, use a server profile exit.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,” on page 33](#).

Note: You should modify the DTCPARMS file for the MRoute server if you need to specify any of the MRoute server input parameters. See [“MPROUTE Command” on page 203](#) for a complete list.

Step 5. Optional: Create static routes

To create IPv4 and IPv6 static routes, use the GATEWAY statement in the TCP/IP server configuration file. For information on the syntax of the GATEWAY statement, see [“GATEWAY Statement” on page 568](#).

During initialization, MRoute learns of static routes by reading the internal routing table set up by TCP/IP. If static routes are changed during execution by OBEYFILE command processing, MRoute is dynamically notified of the changes by TCP/IP. MRoute will advertise active static routes to other routers if allowed by configuration (for example, the IMPORT_STATIC_ROUTES parameter of the AS_BOUNDARY_ROUTING or IPV6_AS_BOUNDARY_ROUTING configuration statements).

A static route cannot be replaced or modified by MRoute, even if a better dynamic route can be learned and even if the static route is not actually available (but a static route that is not available will not be advertised by MRoute). Because of this, the use of static routes with MRoute is not recommended unless it is to provide routing over an interface over which no routing protocol is being communicated.

Step 6. Optional: Configure OSPF authentication if using the IPv4 OSPF protocol

MRoute supports defining the IPv4 OSPF authentication type by area or by interface. All interfaces attached to an area default to the type of authentication defined for that area on the AREA configuration statement, unless overridden on the OSPF_INTERFACE configuration statement. The values of authentication keys must be defined on OSPF_INTERFACE statements in any case. All routers which could become neighbors of each other must use the same authentication type and key, or OSPF communication between the routers will not be possible.

Virtual links behave similarly to interfaces for authentication purposes. A virtual link will default to use the same type of authentication that is specified for the backbone area unless overridden on the VIRTUAL_LINK configuration statement. When the authentication type is not NONE, the value of the authentication key must be specified on the VIRTUAL_LINK configuration statement. There is no requirement for a virtual link to have the same authentication key value as its underlying real interface.

OSPF authentication does not protect the contents of an OSPF packet. These packets are not encrypted. However, it does provide verification that the packet is genuine.

Tip: Unlike IPv4 OSPF, IPv6 OSPF does not provide its own, protocol-layer authentication. It relies instead on authentication provided by IPv6 IPsec.

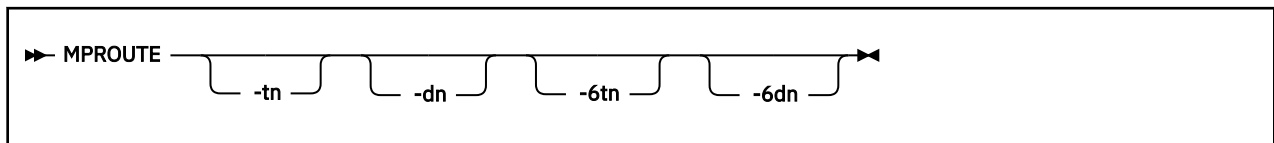
There are two methods of IPv4 OSPF authentication: password and MD5 cryptographic.

Password authentication is very basic: an 8-byte password is appended to all OSPF packets and sent in the clear with the rest of the packet. If the sent password matches that defined by the packet receiver, the packet is accepted.

MD5 authentication is more sophisticated. The combination of the OSPF packet and the MD5 key is summarized into a 16-byte message digest, which is appended to the packet and sent. The keys are never sent, only the message digests. The receiver then attempts to recreate the message digest from the combination of its defined key and the OSPF packet. If the digest is successfully recreated, the packet is accepted; otherwise, it is rejected. MD5 authentication also contains a monotonic increasing counter to protect against replay attacks.

If MD5 cryptographic authentication is being used, a 16-byte MD5 key must be defined on the `OSPF_INTERFACE` configuration statement.

MPROUTE Command



Purpose

The MPROUTE command accepts the command line parameters listed below. These parameters are valid when starting the program from either the CMS command line or through DTCPARMS.

Operands

-tn

External tracing level for MPRoute initialization and IPv4 routing protocols, where n is a supported trace level. The following values are supported:

1. Informational messages
2. Formatted packet trace and informational messages

-dn

Internal debugging level for MPRoute initialization and IPv4 routing protocols, where n is a supported debug level. This parameter is intended for service, as it provides information needed for debugging problems.

-6tn

External tracing level for IPv6 routing protocols, where n is a supported trace level. The following values are supported:

1. Informational messages
2. Formatted packet trace and informational messages

-6dn

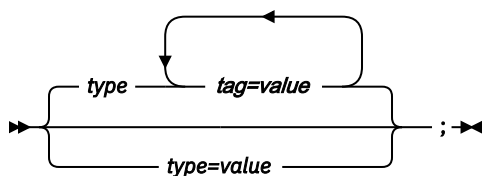
Internal debugging level for IPv6 routing protocols, where n is a supported debug level. This parameter is intended for service, as it provides information needed for debugging problems.

Usage Note

For more information on the trace and troubleshooting problems with the MPRoute server, see the "MPRoute Traces and Debug Information" in *z/VM: TCP/IP Diagnosis Guide*.

MPRoute configuration file

Statements in the MPRoute configuration file have the following syntax:



where

type

Specifies what is to be configured.

tag=value

Specifies a parameter and its associated value.

type=value

Is used for statements that have only a single parameter.

The following are the syntax rules for the MPRoute configuration statements:

- Types, tags, and values can be specified in mixed case.
- Every configuration statement must end with a semicolon (;).
- Blanks and comments are supported. Comments are identified by a semicolon in any column. Comments cannot appear within a configuration statement.
- Statements may begin in any column.
- There must be no sequence numbers in the data set or file.
- Statements with only optional operands must have at least one operand coded, even if all operands have defaults.

A sample MPRoute configuration file is shipped as MPROUTE SCONFIG on TCPMAINT's 591 disk.

INCLUDE

Purpose

If you are configuring a complex environment, you can group related statements into separate files and use the INCLUDE statement to include them in your MPRoute configuration file. This statement causes configuration statements from the specified file to be included at the point that the INCLUDE statement is encountered in the configuration file.

Format

```
➤ INCLUDE — fn ft ➤
```

Parameters

fn ft

The file name and file type of the file to be included.

Rules:

- An INCLUDE statement must be the only configuration statement on the line.
- An INCLUDE statement must be on a single line. It cannot span multiple lines.
- There can be no more than ten nested INCLUDE statements.
- Only one INCLUDE statement can be specified per line. Anything that follows the INCLUDE statement is ignored.

- The system uses the first *fn ft* file found in the CMS search order.
- If the file cannot be found or opened, the INCLUDE statement is ignored and MPRoute continues to run.

Note: If a syntax error is encountered in the final version of the configuration file after one or more INCLUDE files were processed, use debug level d1 to print a copy of the expanded configuration file to your MPRoute trace. This helps identify the correct line number where the syntax error was found, as reported in the error message.

Creating the MPRoute configuration file

Follow these steps to configure OSPF and RIP (IPv4 and IPv6):

1. Set the OSPF router ID, if the OSPF protocol is used. For details, see [“Notes on setting the OSPF router ID” on page 205](#).
2. Define OSPF areas, if the OSPF protocol is used. For details, see [“Notes on defining OSPF areas” on page 205](#).
3. Limit information exchange between OSPF areas, if the OSPF protocol is used. For details, see [“Notes on limiting information exchange between OSPF areas” on page 206](#).
4. Define IPv4 interfaces, if the IPv4 OSPF or IPv4 RIP protocol is used. For details, see [“Notes on defining IPv4 interfaces” on page 207](#).
5. Define IPv6 interfaces, if the IPv6 OSPF or IPv6 RIP protocol is used. For details, see [“Notes on defining IPv6 interfaces” on page 212](#).
6. Define interface costs (OSPF_INTERFACE, RIP_INTERFACE, IPV6_OSPF_INTERFACE, and IPV6_RIP_INTERFACE). For details, see [“Notes on defining interface costs” on page 214](#).
7. Configure virtual links, if the OSPF protocol is used. For details, see [“Notes on configuring virtual links” on page 215](#).
8. Manage high-cost links, if the OSPF protocol is used. For details, see [“Notes on managing high-cost links” on page 215](#).
9. Define RIP filters, if the RIP protocol is used. For details, see [“Notes on defining RIP filters” on page 216](#).
10. Define route precedence in a multiprotocol environment, if the OSPF protocol is used. For details, see [“Notes on defining route precedence in a multiprotocol environment” on page 216](#).

Notes on setting the OSPF router ID

Every router in an OSPF autonomous system must be assigned a unique router ID.

IPv4 OSPF

The ROUTERID parameter of the OSPF configuration statement should be coded within the MPRoute configuration file to assign the router ID. The value must be the IP address of one of the OSPF_INTERFACES defined in the MPRoute configuration file. If the ROUTERID parameter of the OSPF configuration statement is not coded, MPRoute chooses the IP address from one of the OSPF_INTERFACE statements as the router ID.

IPv6 OSPF

The ROUTERID parameter of the IPV6_OSPF configuration statement is coded within the MPRoute configuration file to assign the IPv6 OSPF router ID. This router ID is any IPv4-style dotted-decimal value except for 0.0.0.0, with care taken to assure its uniqueness across routers within the IPv6 autonomous system. If the ROUTERID parameter of the IPV6_OSPF configuration statement is not coded and the IPv4 OSPF protocol is also being used, MPRoute will use the IPv4 OSPF router ID as the IPv6 OSPF router ID. If the IPv4 OSPF protocol is not being used, the ROUTERID parameter of the IPV6_OSPF statement must be specified.

Notes on defining OSPF areas

OSPF areas are collections of contiguous IP subnetworks. The function of areas is to reduce the OSPF overhead required to compute routes to destinations in different areas. Overhead is reduced because less

information is exchanged and stored by routers and because fewer CPU cycles are required for a less complex route table calculation.

Every OSPF AS that will have multiple areas must have at least a backbone area. The backbone is always identified by area number 0.0.0.0. For networks with multiple areas, the backbone provides a core that connects the areas. Unlike other areas, the backbone's subnets can be physically separate. In this case, logical connectivity of the backbone is maintained by configuring virtual links between backbone routers across intervening non-backbone areas. For more information on this subject, see [“Notes on configuring virtual links” on page 215](#).

Routers that attach to more than one area function as area border routers. All area border routers are part of the backbone, so they must either attach directly to a backbone IP subnet or be connected to another backbone router over a virtual link.

The information and algorithms used by OSPF to calculate routes vary according to whether the destination is within the same area, in a different area within the OSPF AS, or external to the OSPF AS. Every router maintains a database of all links within its area. A shortest path first algorithm is used to calculate the best routes to destinations within the area from this database. Routes between areas are calculated from summary advertisements originated by area border routers for destinations located in other areas of the OSPF AS. External routes (for example, routes to destinations that lie within a RIP AS) are calculated from AS external advertisements originated by AS boundary routers and flooded throughout the OSPF AS.

IPv4 OSPF

Use the AREA configuration statement to define the areas to which a router attaches. If you do not use the AREA statement, the default is that all OSPF interfaces attach to the backbone area.

IPv6 OSPF

Use the IPV6_AREA configuration statement to define the areas to which a router attaches. If you do not use the IPV6_AREA statement, the default is that all IPv6 OSPF interfaces attach to the backbone area.

Notes on limiting information exchange between OSPF areas

When area border routers are configured, the OSPF route information that crosses the area boundary can be controlled using configuration statements.

One option is to define an area as a stub area. AS external advertisements are never flooded into stub areas. In addition, the origination into the stub area of summary advertisements for interarea routes can be suppressed, creating what is commonly known as a totally stubby area.

Destinations external to the stub area are still reachable due to the area border routers advertising default routes into stub areas. Traffic within the stub area for unknown destinations is forwarded to the area border router (using the default route). It is acceptable for routers within the area to use a default route for traffic destined outside the AS. The border router uses its more complete routing information to forward the traffic on an appropriate path toward its destination.

The following requirements must be met for an area to be defined as a stub area:

- No virtual links are configured through the area to maintain backbone connectivity.
- No routers within the area are AS boundary routers (OSPF routers that advertise routes from external sources as AS external advertisements).

Tip: Static routes and RIP routes are AS external.

Another option is to use IP address ranges to limit the number of summary advertisements originated out of an area. In IPv4, a range is defined by an IP address and an address mask. Destinations are considered to fall within the range if the destination address and the range IP address match after the range mask has been applied to both addresses. In IPv6, a range is defined by an IP address prefix and a prefix length, and destinations are considered to fall within the range if a destination address and a range IP address match for the length of the range's prefix.

When a range is configured for an area at an area border router, the border router suppresses summary advertisements for destinations within that area that fall within the range. The suppressed

advertisements would have been originated into the other areas to which the border router attaches. Instead, the area border router may originate a single summary advertisement for the range or no advertisement at all, depending on the option chosen with the configuration statement.

Rules:

1. If the range is not advertised, there will be no interarea routes for any destination that falls within the range.
2. Ranges cannot be used for areas through which virtual links are configured to maintain backbone connectivity.

IPv4 OSPF

The following statement configures an OSPF area as a stub area. The `Import_Summaries=No` parameter will result in the suppression of summary advertisements for interarea routes into the stub area, creating a totally stubby area:

```
AREA
  Area_Number=2.2.2.2
  Stub_area=Yes
  Import_Summaries=No;
```

IPv6 OSPF

The following statement configures an IPv6 OSPF area as a stub area. The `Import_Summaries=No` parameter will result in the suppression of summary advertisements for interarea routes into the stub area, creating a totally stubby area:

```
IPV6_AREA
  Area_Number=1.1.1.1
  Stub_area=Yes
  Import_Prefixes=No;
```

The following `IPV6_RANGE` statement causes MPRoute to advertise all destinations in the 2001:0DB8:0:31::/64 prefix from area 6.6.6.6 to the backbone area (Area 0.0.0.0) as a single route to the 2001:0DB8:0:31::/64 prefix:

```
IPV6_RANGE
  Prefix=2001:0DB8:0:31::/64
  Area_Number=6.6.6.6
  Advertise=Yes;
```

Notes on defining IPv4 interfaces

Each interface in use by the stack should be defined to MPRoute using an `OSPF_INTERFACE`, `RIP_INTERFACE`, or `INTERFACE` statement. This substep describes the differences between interface types that you should consider when configuring interfaces to MPRoute. In general, use the following guidelines:

- An interface over which the OSPF protocol is communicated with other routers must be configured with the `OSPF_INTERFACE` statement.
- An interface over which the RIP protocol is communicated with other routers must be configured with the `RIP_INTERFACE` statement.
- Either all other interfaces should be configured with the `INTERFACE` statement, or MPRoute should be configured to ignore undefined interfaces using the `IGNORE_UNDEFINED_INTERFACES` parameter of the `GLOBAL_OPTIONS` statement in the MPRoute configuration file. It is important that one or the other be done.

If MPRoute is not configured to ignore undefined interfaces, it configures stack interfaces that are not defined to MPRoute with default values. These values might not be desirable. For example, the class mask is used as the subnet mask and 576 is used as the MTU value. Furthermore, MPRoute overrides stack values with its default values unless the stack's value for the MTU has been specified on the `LINK` statement for an interface. To prevent these situations, either configure every interface, even those that are not using RIP or OSPF, or configure MPRoute to ignore undefined interfaces.

A VIPA interface is an exception to these guidelines and is discussed in more detail later in this substep.

TCP/IP enforces RFC rules against using either a subnetwork's broadcast or network address as a host address. (An address that has all ones in the host portion is a subnet broadcast address. An address that has all zeros in the host portion is the subnet's network address.) Therefore, the `subnet_mask` on an `OSPF_INTERFACE`, `RIP_INTERFACE`, or `INTERFACE` statement should have enough zero bits such that no home address in that subnet has all zeros or all ones in the host portion of the address. For example if a subnet has two home addresses 10.1.1.1 and 10.1.1.2, the subnet mask must have zeros in at least two bits; for example, 255.255.255.252. However, if a subnet has four home addresses 10.1.1.1, 10.1.1.2, 10.1.1.3, and 10.1.1.4, the subnet mask must have zeros in at least three bits; for example, 255.255.255.248. In this case, there could be up to 6 home addresses in that subnet (10.1.1.1 through 10.1.1.6). In general, if a subnet mask has n zero bits, then there can be up to $((2^n)-2)$ home addresses in that subnet. This limit applies even if the home addresses are configured on different TCP/IP stacks.

Rules:

1. MPRoute supports a maximum of 255 real, physical, IPv4 interfaces (that is, interfaces on which data can actually be sent and received). There is no theoretical limit on how many VIPAs can be configured, though there are practical limits imposed by network design.
2. RIP version 1 uses broadcast and RIP version 2 uses multicast. RIP version 1 will not run on a medium that supports multicast but not broadcast, so the OSA-Express microcode level must support broadcast. Some levels of OSA-Express microcode support multicasting but not broadcasting. In this case, RIP version 2, which relies on multicast, is recommended over RIP version 1, which relies on broadcast.

Configuring multi-access parallel interfaces: Whenever configuring multi-access parallel interfaces (primary and backup redundant interfaces having IP addresses in the same network) for MPRoute (OSPF), use the `Parallel_OSPF` parameter on the `OSPF_INTERFACE` statement to designate whether each OSPF interface is primary or backup. If the `IP_address` parameter on an `OSPF_INTERFACE` statement is using wildcards (*), also include the `Name` parameter for the interface to distinguish the primary from the backups. For additional information on the `OSPF_INTERFACE` statement, see [“OSPF_INTERFACE” on page 223](#).

Point-to-point (For example CTC): For point-to-point interfaces, the destination IP address must be known to MPRoute. If OSPF or RIP is run on the interface, the destination IP address is learned when the router at the other end comes up and shares information with MPRoute. Additionally, defining a destination address on an interface that is running OSPF or RIP allows MPRoute to learn and advertise a route to that destination address before a neighboring router has become fully adjacent. This can be beneficial if the neighboring router takes a long time to come up, or is otherwise not expected to be promptly and reliably available.

Tip: For a point-to-point interface running OSPF, MPRoute does not advertise a host route to its own home address on the point-to-point interface. By default, MPRoute advertises a host route to the link destination, and relies on the router at the other end to advertise a host route to MPRoute's home address. This behavior is described by the OSPF architecture, RFC 1583 (OSPF version 2), section 12.4.1. This means that MPRoute's home address on the point-to-point link will not be advertised as reachable unless there is a neighboring router available to advertise it. MPRoute implements an extension described in RFC 2328 to overcome this limitation. If a neighboring router will not be reliably available over the point-to-point link, you might want to code the parameter `SUBNET=YES` on the `OSPF_INTERFACE` statement for the point-to-point interface. This causes MPRoute to implement option 2 described in RFC 2328, section 12.4.1.1, and advertise a route to the point-to-point link's subnet address. This makes both endpoints reachable regardless of the status of a neighboring router.

If the interface is simply an `INTERFACE` (not running OSPF or RIP), specify the `DESTINATION_ADDR` parameter to allow for the creation of a host route to the address at the remote end of the interface.

Sample `OSPF_INTERFACE`:

```
OSPF_INTERFACE
  IP_Address=9.67.106.7
  Name=CTC7T04
  Subnet_mask=255.255.255.0
```

```
Attaches_to_Area=1.1.1.1
Destination_Addr=9.67.106.4;
```

Sample RIP_INTERFACE:

```
RIP_INTERFACE
  IP_Address=9.67.103.7
  Name= CTC7T06
  Subnet_mask=255.255.255.0
  Destination_Addr=9.67.103.6
  RIPV2=Yes;
```

Sample INTERFACE:

```
INTERFACE
  IP_Address=9.67.111.1
  Name=CTCX
  Subnet_mask=255.255.255.0
  Destination_addr=9.67.111.2;
```

Non-broadcast network interfaces: If the OSPF or RIP protocol communicates with one or more routers over a non-broadcast network interface, MPRoute must know the IP addresses of the other routers (neighbors) with which it needs to communicate. For non-broadcast network interfaces, there is no underlying signaling that allows the stack to learn the required IP addresses. As a result, the neighbor addresses must be configured to MPRoute with the parameters configured as follows:

- DR_NEIGHBOR and/or the NO_DR_NEIGHBOR parameters on the OSPF_INTERFACE statement
- NEIGHBOR parameter on the RIP_INTERFACE statement
- NON_BROADCAST=YES and ROUTER_PRIORITY parameters on the OSPF_INTERFACE statement

In the OSPF case, DR_NEIGHBOR defines which routers within the non-broadcast network can become the designated router. NO_DR_NEIGHBOR defines which routers cannot become the designated router. ROUTER_PRIORITY defines the priority of this router on the non-broadcast network so that the designated router can be elected for the network. Note that multiple DR_NEIGHBOR and NO_DR_NEIGHBOR parameters can be coded on one statement.

Sample OSPF_INTERFACE:

```
OSPF_INTERFACE
  IP_Address=9.37.84.49
  Name=HCHE00
  Subnet_mask=255.255.255.0
  Attaches_to_Area=1.1.1.1
  Non_Broadcast=Yes
  DR_Neighbor=9.37.84.53
  No_DR_Neighbor=9.37.84.63
  Cost0=3
  Router_Priority=2;
```

Sample RIP_INTERFACE:

```
RIP_INTERFACE
  IP_Address=9.37.104.79
  Name=ATME00
  Subnet_mask=255.255.255.0
  RIPV2=Yes
  Neighbor=9.37.104.85
  Neighbor=9.37.104.53;
```

Sample INTERFACE:

```
INTERFACE
  IP_Address=9.77.13.49
  Name=ATMB00
  Subnet_mask=255.255.255.0;
```

Broadcast network interfaces: When the OSPF or RIP protocol is communicated over a broadcast medium such as Ethernet, these networks allow for broadcasting and multicasting. Therefore, it is not necessary for MPRoute to know the IP addresses of the other routers on the network for OSPF or RIP

packets to be communicated with those routers. MPRoute sends packets to the other routers on the network by using appropriate broadcast or multicast addresses. The IP addresses of the other routers are learned as OSPF/RIP packets are received from them. The OSPF_INTERFACE must include the ROUTER_PRIORITY parameter to assist in electing a designated router for the network.

Sample OSPF_INTERFACE:

```
OSPF_INTERFACE
  IP_Address=9.59.101.5
  Name=LINK1
  Subnet_mask=255.255.255.0
  Attaches_to_Area=1.1.1.1
  Cost0=2
  Router_Priority=1;
```

Sample RIP_INTERFACE:

```
RIP_INTERFACE
  IP_Address=9.29.107.3
  Name=LINK2
  Subnet_mask=255.255.255.0
  RIPV2=Yes;
```

Sample INTERFACE:

```
INTERFACE
  IP_Address=9.77.14.49
  Name=ETHB00
  Subnet_mask=255.255.255.0;
```

For interfaces into broadcast media which contain routers that do not support multicast, it is possible to configure the interfaces as non-broadcast network interfaces. This would cause MPRoute to unicast to the neighbor addresses rather than using a multicast address. However, it would also be necessary to configure all the routers on the network to unicast. Otherwise, their multicast packets would never be received.

Note that it is possible to define neighbors using DR_NEIGHBOR and/or NO_DR_NEIGHBOR parameters for OSPF_INTERFACES and using NEIGHBOR parameters for RIP_INTERFACES that are broadcast capable, but it is not required or recommended. If you define neighbors on these interfaces, you must define all of them, as MPRoute will not communicate RIP or OSPF to undefined neighbors if any are defined on an interface.

VIPA interfaces: If only the RIP protocol is used by MPRoute, VIPA interfaces should be defined with the INTERFACE statement. If only OSPF or if both OSPF and RIP are used by MPRoute, VIPA interfaces should be defined with the OSPF_INTERFACE statement.

Sample OSPF_INTERFACE:

```
OSPF example:
  OSPF_INTERFACE
  IP_Address=4.4.4.4
  Name=VIPA1
  Subnet_mask=255.255.255.252;
```

Sample INTERFACE:

```
non-OSPF example:
  INTERFACE
  IP_Address=6.6.6.6
  Name=VIPA1
  Subnet_mask=255.255.255.252;
```

Rule: The most specific subnet mask you can specify is 255.255.255.252.

If the name in an OSPF_INTERFACE or INTERFACE statement refers to a link of type VIRTUAL, then MPRoute generates and advertises the following routes whenever applicable:

- A network route to the network specified in that statement

- A subnet route to the subnet specified in that statement
- A host route to the IP_address specified in that statement

Following are the conditions for advertising these routes on a physical network interface to a network:

- Network route - if VIPA is not in the same network as the physical network interface and is allowed by filters or RANGE.
- Subnet route - VIPA subnet routes are advertised in MPRoute in all conditions, except for RIP when filters prevent it.
- Host route - as allowed by filters or RANGE. Advertisement of the host route for a VIPA defined on an OSPF_INTERFACE statement can be controlled by the SUBNET parameter on the OSPF_INTERFACE statement that defines that VIPA. If SUBNET=YES, then the host route is not advertised. If SUBNET=NO (the default), the host route is advertised. Take care in using this parameter. VIPA host routes should not be suppressed for VIPAs whose subnet might exist on multiple hosts. It is up to the user to ensure these restrictions are enforced, as they are not and cannot be enforced by MPRoute.

On the RIP_INTERFACE statement for a physical network interface, the VIPA routes are allowed to be advertised by the following filter parameters:

- Send_Net_Routes
- Send_Subnet_Routes
- Send_Host_Routes, and Send_Only

In addition, the global FILTER and Send_Only statements for RIP can be used to specify which routes are advertised or not.

For OSPF, the RANGE statement can be used to advertise or not to advertise the VIPA routes external to an area in terms of address range based on a subnet mask.

Note: For RIP, the Send_Only = (VIRTUAL) filter in conjunction with the Send_Net_Routes, Send_Subnet_Routes, and Send_Host_Routes filters, or the FILTER statement with VIPA routes, indicates whether or not VIPA routes can be advertised over a RIP interface. Unlike RIP, there are no routing filters for OSPF. For OSPF, the RANGE statement can be used to control which address range of routes can be advertised or not advertised external to an area; however, it is not granular enough for use as a routing filter. In area-border router configurations, if there are multiple VIPA addresses that are uniquely subnetted, the RANGE statement can be used to specify which VIPA subnet address range of routes can be advertised or not advertised external to an area.

Method of assigning interface definitions to stack interfaces (wildcard and explicit): Wildcard interface definitions can be a convenient way of making your interface definitions easier. However, to avoid unintended results, be sure to understand how they are parsed, and how different types of interface definitions interact with each other. Following is the outline of the algorithm MPRoute uses to find the matching definitions in the MPRoute configuration file for an IPv4 stack interface.

1. Search for a RIP_Interface definition for the interface as follows:
 - a. Search for an explicit matching RIP_Interface definition. If found, use that definition and go to step [“2” on page 212](#).
 - b. Search for a one-octet wildcard RIP_Interface definition (n.n.n.*), where the first three octets match the first three octets of the interface's home address and the name matches the interface's link name. If found, use that definition and go to step [“2” on page 212](#).
 - c. Search for a two-octet wildcard RIP_Interface definition (n.n.*.*), where the first two octets match the first two octets of the interface's home address and the name matches the interface's link name. If found, use that definition and go to step [“2” on page 212](#).
 - d. Search for a three-octet wildcard RIP_Interface definition (n.*.*.*), where the first octet matches the first octet of the interface's home address and the name matches the interface's link name. If found, use that definition and go to step [“2” on page 212](#).
 - e. Search for a one-octet wildcard RIP_Interface definition (n.n.n.*), where the first three octets match the first three octets of the interface's home address, ignoring the name parameter if coded. If found, use that definition and go to step [“2” on page 212](#).

- f. Search for a two-octet wildcard RIP_Interface definition (n.n.*.*), where the first two octets match the first two octets of the interface's home address, ignoring the name parameter if coded. If found, use that definition and go to step “2” on page 212.
- g. Search for a three-octet wildcard RIP_Interface definition (n.*.*.*), where the first octet matches the first octet of the interface's home address, ignoring the name parameter if coded. If found, use that definition and go to step “2” on page 212.
- h. If there is a four-octet wildcard RIP_Interface definition (*.*** or ALL), use that definition and go to step “2” on page 212.

Restriction: Only one four-octet wildcard of each type (OSPF_INTERFACE or RIP_INTERFACE) is allowed.

2. Search for an OSPF_Interface definition for the interface. Note that this step is done regardless of the outcome of step “1” on page 211. The steps for searching OSPF_Interface definitions are the same as the steps for searching RIP_Interface definitions described above, except that OSPF_Interface definitions are searched.
3. If either a RIP_Interface or an OSPF_Interface definition, or both, are found, the algorithm is complete. In this case, Interface definitions are not searched. If neither a RIP_Interface nor an OSPF_Interface definition was found, go to step “4” on page 212.
4. Search for an Interface definition for the interface. The steps for searching Interface definitions are the same as the steps for searching RIP_Interface statements described above, except that Interface definitions are searched.
5. If no definitions are found, check the value of Global_Options Ignore_Undefined_Interfaces. If this option is turned on, the interface is ignored. If it is not turned on, the interface is treated as if it were defined with an INTERFACE statement, with an MTU value of 576 and a subnet mask of the class mask.

The algorithm is complete. Key conclusions of this algorithm are as follows:

- A wildcard interface definition with a matching name is preferred over a wildcard interface definition of the same type without a matching name, regardless of which definition is a more specific wildcard.
- If a RIP_Interface or an OSPF_Interface definition, or both, are found, Interface definitions are not considered. This means that any matching RIP_Interface or OSPF_Interface definition supersedes all Interface definitions, even if the Interface definitions are explicit or are more specific wildcard matches. For example, a three-octet wildcard OSPF_Interface definition supersedes an explicit Interface definition.
- An interface can be both a RIP_Interface and an OSPF_Interface. MPRoute supports running both protocols over the same interface. However, an interface cannot be both an interface that runs no routing protocol (that is, defined with the INTERFACE statement) and one that runs RIP, OSPF, or both.

Notes on defining IPv6 interfaces

Each IPv6 interface in use by the stack can be defined to MPRoute using an IPV6_OSPF_INTERFACE, IPV6_RIP_INTERFACE, or IPV6_INTERFACE statement. Defining IPv6 interfaces to MPRoute is much simpler than defining IPv4 interfaces, because you do not specify IP addresses or MTU sizes to MPRoute. Instead, you simply define interfaces by their names and MPRoute learns IP addresses and MTU sizes from the TCP/IP stack. Also, because multicast support is a basic requirement of IPv6, there are no non-broadcast multiaccess considerations or other considerations that might require definitions of neighbors or destination addresses.

Use the following guidelines when configuring IPv6 interfaces to MPRoute:

- An interface over which the IPv6 OSPF protocol is communicated with other routers must be configured with the IPV6_OSPF_INTERFACE statement.
- An interface over which the IPv6 RIP protocol is communicated with other routers must be configured with the IPV6_RIP_INTERFACE statement.
- All other interfaces can be configured with the IPV6_INTERFACE statement, if MPRoute default values are not acceptable or you choose to define additional prefixes on the interface.

- The interface name must be coded on the IPV6_INTERFACE, IPV6_OSPF_INTERFACE, and IPV6_RIP_INTERFACE statements. The value of the NAME parameter must match the interface name coded on the LINK statement in the TCP/IP configuration file. Wildcard names ending with an asterisk (*) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.
- Numeric interface names containing a decimal point (for example, 123.456) are not allowed. If you have configured this type of interface name in your TCP/IP configuration file, you must change the name there.
- To define one or more prefixes on an interface, use the PREFIX parameter on the IPV6_OSPF_INTERFACE, IPV6_RIP_INTERFACE, and IPV6_INTERFACE statements. You should only need to do this for prefixes that you need to define to an interface, which will not be learned using IPv6 router discovery. Also note that prefixes defined to MPRoute in this manner are not used by TCP/IP to autoconfigure home addresses on the interface.

The following sample shows an IPv6 OSPF interface with prefixes defined:

```
IPV6_OSPF_INTERFACE
NAME=OSQDIO4L6
PREFIX=2001:0DB8:1::/48
PREFIX=2001:0DB8:2::/48;
```

The prefixes defined in this manner on an IPv6 OSPF interface are advertised as reachable, and are also included in the link LSA generated by MPRoute, so all IPv6 OSPF routers on the link will know they are local prefixes. If MPRoute is also running IPv6 RIP, they are also advertised into the IPv6 RIP autonomous system as reachable, if IPv6 RIP filters permit it.

The following sample shows an IPv6 RIP interface with prefixes defined:

```
IPV6_RIP_INTERFACE
NAME=OSQDIO3L6
PREFIX=2001:0DB8:3::/48
PREFIX=2001:0DB8:4::/48;
```

The prefixes defined in this manner on an IPv6 RIP interface are advertised into the IPv6 RIP autonomous system as reachable if IPv6 RIP filters permit it. They are also advertised into the IPv6 OSPF autonomous system as reachable, if MPRoute is running IPv6 OSPF and is configured as an IPv6 AS boundary router importing RIP routes.

The following sample shows an IPv6 generic interface with prefixes defined:

```
IPV6_INTERFACE
NAME=OSQDIO2L6
PREFIX=2001:0DB8:5::/48
PREFIX=2001:0DB8:6::/48;
```

The prefixes defined in this manner on an IPv6 generic interface are advertised into the RIP autonomous system as reachable, if MPRoute is running IPv6 RIP and IPv6 RIP filters permit it. They are also advertised into the IPv6 OSPF autonomous system as reachable, if MPRoute is running IPv6 OSPF and is configured as an IPv6 AS boundary router importing direct routes.

Method of assigning interface definitions to stack interfaces (wildcard and explicit): For IPv6 interfaces, interface-name wildcards can be used to simplify definitions. However, be sure to understand how they are parsed, and how different types of interface definitions interact with each other, to avoid unintended results. Following is the outline of the algorithm MPRoute uses to find the matching definitions in the MPRoute configuration file for an IPv6 stack interface.

1. Search for an IPV6_RIP_Interface definition for the interface as follows:
 - a. Search for an explicit matching IPV6_RIP_Interface statement for the interface. This is one where the name parameter exactly matches the interface's name. If one is found, use that definition and go to step b.
 - b. Search for the best IPV6_RIP_Interface wildcard match for the name. The IPV6_RIP_Interface wildcard definitions are searched, starting with the most specific (longest wildcard name string)

and checking each in order of declining specificity until a match is found. As soon as a match is found, use that definition and go to step b.

2. Search for an IPv6_OSPF_Interface definition for the interface. Note that this step is done regardless of the outcome of step a. The steps for searching IPv6_OSPF_Interface definitions are the same as the steps for searching IPv6_RIP_Interface definitions described above, except that IPv6_OSPF_Interface definitions are searched.
3. If either an IPv6_RIP_Interface or an IPv6_OSPF_Interface definition, or both, are found, the algorithm is complete. In this case, IPv6_Interface definitions are not searched. If neither an IPv6_RIP_Interface nor an IPv6_OSPF_Interface definition was found, go to step d.
4. Search for an IPv6_Interface definition for the interface. The steps for searching IPv6_Interface definitions are the same as the steps for searching IPv6_RIP_Interface statements described above, except that IPv6_Interface definitions are searched.
5. If no definitions are found, check the value of Global_Options Ignore_Undefined_Interfaces. If this option is turned on, the interface is ignored. If it is not turned on, the interface is treated as if it were defined with an IPv6_Interface statement. Default values will be used for all parameters.

The algorithm is complete. Key conclusions of this algorithm are as follows:

- If an IPv6_RIP_Interface definition, an IPv6_OSPF_Interface definition, or both, are found, IPv6_Interface definitions are not considered. This means that any matching IPv6_RIP_Interface or IPv6_OSPF_Interface definition supersedes all IPv6_Interface definitions, even if the IPv6_Interface definitions are explicit or more specific wildcard matches. For example, an IPv6_OSPF_Interface definition with a name parameter of V* supersedes any IPv6_Interface, explicit or wildcard, with a name parameter that begins with V. In this case, the IPv6_Interface definition is redundant and will never be used. If MPRoute detects this case, it issues message EZZ8068I and deletes the redundant IPv6_Interface definition.

Note: If an IPv6_Interface definition has already been selected for an interface that is installed in the stack, and then an IPv6_OSPF_Interface or IPv6_RIP_Interface definition that would make that IPv6_Interface definition redundant is added using RECONFIG, MPRoute issues message EZZ8069I and retains the IPv6_Interface definition.

- An interface can be both an IPv6_RIP_Interface and an IPv6_OSPF_Interface. MPRoute supports running both protocols over the same interface. However, an interface cannot be both an interface that runs no routing protocol (that is, defined with the IPv6_Interface statement) and one that runs IPv6 RIP, IPv6 OSPF, or both.

Notes on defining interface costs

Both the OSPF and RIP protocols have a cost value associated with interfaces. With both protocols, the cost of a route to reach a destination is the sum of the costs of each link that will be traversed on the way to the destination.

The method for configuring cost values differs between the OSPF and RIP protocols. The cost values of OSPF links should be configured to ensure that preferred routes to destinations will have a lower cost than less preferable routes. The less preferable routes, with the higher cost, will not be used except upon failure of the preferred routes.

The reasons for preferring one route over another are numerous. One approach for assigning OSPF link costs would be to set the costs to values inversely proportional to the bandwidth of the physical media. This would result in higher bandwidth routes having lower costs, thus becoming the preferred routes.

The cost values of RIP links are generally set to a value of 1. This results in the cost of a route to a destination being the number of hops to reach the destination.

IPv4 OSPF and RIP

The cost value of an OSPF interface is set using the COST0 parameter of the OSPF_INTERFACE statement. The in metric and out metric of a RIP interface are set using the IN_METRIC and OUT_METRIC parameters of the RIP_INTERFACE statement.

IPv6 OSPF and RIP

The cost value of an IPv6 OSPF interface is set using the COST parameter of the IPV6_OSPF_INTERFACE statement. The in metric and out metric of an IPv6 RIP interface are set using the IN_METRIC and OUT_METRIC parameters of the IPV6_RIP_INTERFACE statement.

Notes on configuring virtual links

The OSPF protocol is dependent upon complete connectivity of the backbone area. To maintain backbone connectivity, each backbone router must be interconnected. If the configuration of an OSPF autonomous system is such that the backbone area will become separated into two or more disconnected sections, connectivity must be restored for the protocol to work correctly. This can be done using a virtual link. An OSPF virtual link should not be confused with a VIPA link. Virtual links can be configured between any two backbone routers that have an interface to a common non-backbone area.

IPv4 OSPF

The VIRTUAL_LINK statements specify the router ID of the link endpoint and must be configured at both endpoints.

IPv6 OSPF

The IPV6_VIRTUAL_LINK statements specify the router ID of the link endpoint and must be configured at both endpoints.

Notes on managing high-cost links

The periodic nature of OSPF routing traffic requires a link's underlying data-link connection to be constantly open. This can result in unwanted usage charges on network segments whose costs are very high. There are two configuration steps that can be taken to inhibit the periodic nature of the protocol.

The first step that can be taken is to define the link as a demand circuit. When this is done, link state advertisements (LSAs) sent over the interface will not be periodically refreshed. Only LSAs with real changes will be readvertised. In addition, aging of these LSAs will be disabled such that they will not age out of the link state database.

Another step that can be taken is to define hello suppression for the link. Hello suppression is only meaningful if the link is a demand circuit and is point-to-point. Hello suppression will inhibit the periodic transmission of OSPF hello packets.

IPv4 OSPF

To define OSPF interfaces as demand circuits, the Demand_Circuit=YES parameter must first be specified on the global OSPF configuration statement. Then, the OSPF_INTERFACE statement for each interface to be configured as a demand circuit must be specified with the Demand_Circuit=YES parameter. Use the Hello_Suppression parameter of the OSPF_INTERFACE statement to configure hello suppression. For more information on configuring the Hello_Suppression parameter, see [“OSPF_INTERFACE” on page 223](#). If hello suppression is implemented, the PP_Poll_Interval parameter of the OSPF_INTERFACE statement can be used to specify the interval at which MPRoute should attempt to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available.

IPv6 OSPF

To define IPv6 OSPF interfaces as demand circuits, the Demand_Circuit=YES parameter must first be specified on the global IPV6_OSPF configuration statement. Then, the IPV6_OSPF_INTERFACE statement for each interface to be configured as a demand circuit must be specified with the Demand_Circuit=YES parameter. Use the Hello_Suppression parameter of the IPV6_OSPF_INTERFACE statement to configure hello suppression. For more information on configuring the Hello_Suppression parameter on the IPV6_OSPF_INTERFACE statement, see [“IPV6_OSPF_INTERFACE” on page 245](#). If hello suppression is implemented, the PP_Poll_Interval parameter of the IPV6_OSPF_INTERFACE statement can be used to specify the interval at which MPRoute should attempt to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available.

Notes on defining RIP filters

RIP Filters can be configured to MPRoute such that certain RIP routing information will not be broadcast out to other routers and/or accepted from other routers. The filters can be applied to individual RIP interfaces or to all RIP interfaces. When defining a filter, a filter type (sending or receiving) is specified along with values identifying the route information to be filtered. By using filters, an installation can limit the amount of RIP routing information broadcast into the network and/or the amount of RIP routing information maintained by MPRoute. In addition, filters can be used to hide destination addresses from portions of the network.

IPv4 RIP

To configure a filter for an individual RIP interface, use the `FILTER` parameter of the `RIP_INTERFACE` statement. To configure a filter that applies to all IPv4 RIP interfaces, use the global `FILTER` statement.

IPv6 RIP

To configure a filter for an individual IPv6 RIP interface, use the `FILTER` parameter of the `IPV6_RIP_INTERFACE` statement. To configure a filter that applies to all IPv6 RIP interfaces, use the global `IPV6_RIP_FILTER` statement.

Notes on defining route precedence in a multiprotocol environment

Note that this discussion of route precedence is quite complicated. If only the OSPF or IPv6 OSPF routing protocol, or both, are used in your network, route precedence is less of a concern. If, in addition, none of your OSPF or IPv6 OSPF routers are configured as AS boundary routers, the route precedence concern is entirely eliminated. For environments with multiple protocols or AS boundary routers, the following information is provided. Note that in this discussion, RIP is meant to apply to both RIP and IPv6 RIP, OSPF is meant to apply to both OSPF and IPv6 OSPF, and the OSPF configuration statement is meant to apply to both the OSPF statement and the `IPV6_OSPF` statement.

MPRoute applies an order of precedence in choosing between two routes to the same destination that were learned through different routing protocols or using information provided by an OSPF AS boundary router. To describe this order of precedence applied by MPRoute, a few terms must first be defined.

RIP route

A route learned through the RIP protocol. A RIP route is generated using information provided in a RIP packet from a neighboring router.

OSPF internal route

A route learned through the OSPF protocol where the entire path traversed to reach the destination lies within the OSPF autonomous system.

OSPF external route

A route learned through the OSPF protocol where part of the path traversed to reach the destination does not lie within the OSPF autonomous system. The path will leave the autonomous system if it uses information brought into the OSPF autonomous system by an AS boundary router. This information brought into the OSPF AS may be information imported from a different autonomous system (for example, RIP) or information about destinations statically configured on or directly connected to the AS boundary router.

OSPF external routes fall into two categories based upon the setting of the multiprotocol comparison value. If the comparison value is set to `Type1` on the AS boundary router that imports the external information into the OSPF AS, then OSPF external routes generated using this information will be OSPF type 1 external routes. If the comparison value is set to `Type2` on the AS boundary router, then the generated routes will be OSPF type 2 external routes.

Multiprotocol comparison: You can configure this comparison value to allow for the specification of how route costs from different autonomous systems should be treated when they coexist. In MPRoute, you can configure this value using the `COMPARISON` parameter on the OSPF or `IPV6_OSPF` configuration statements. When `COMPARISON=Type1` is configured, the route cost values used within different autonomous systems (for example, the OSPF AS and the RIP AS) are considered comparable. With `COMPARISON=Type2` configured, the route cost values used with the different autonomous systems are considered non-comparable.

The comparison value can be used in several different ways, depending on the function being performed by a router:

- As an AS boundary router, MPRoute uses the comparison value to determine the type of external routes (type 1 or type 2) generated by routers in the OSPF AS using routing information that the AS boundary router imports into the OSPF AS.
- As an AS boundary router, MPRoute also uses the comparison value in determining how route cost values will be assigned when importing routes from the OSPF AS into the RIP AS.
 - When COMPARISON=Type1 is configured (indicating that cost values are comparable), an OSPF route imported into the RIP AS will be advertised with the actual cost of the OSPF route.

Note:

1. An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type1) occurs when the OSPF route to be imported is an OSPF type 2 external route. When this is the case, the route is not advertised into the RIP AS at all.
 2. It is important to remember the requirement that all destinations in the RIP AS must be reachable with a cost no greater than 15. Using COMPARISON=Type1 requires that the cost values of OSPF routes be low. Any destinations in the OSPF AS that can only be reached from the RIP AS with a cost greater than 15 will become unreachable.
- When COMPARISON=Type2 is configured (indicating that cost values are non-comparable), an OSPF route imported into the RIP AS is advertised with a cost of 1. If a router in the RIP AS has two possible routes to a destination, one internal to the RIP AS and another that was imported from OSPF, this approach results in the route imported from OSPF being favored.

Note: An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type2) occurs when the OSPF route to be imported is an OSPF type 2 external route. When this is the case, the route is advertised into the RIP AS with the actual cost of the OSPF type 2 external route.

- As any router that has routing information from different autonomous systems, MPRoute uses the comparison value while choosing between the routes generated using the information from the different autonomous systems. How the comparison value is used in this case is shown in [Table 29 on page 217](#).

Given these definitions, the order of precedence used in choosing between multiple routes to the same destination, which were learned through the different protocols or by using information provided by an OSPF AS boundary router, can be shown in [Table 29 on page 217](#). In [Table 29 on page 217](#), *Source comparison* refers to the setting of the comparison value (using the COMPARISON parameter on the OSPF configuration statement) on the router that is using the order of precedence to choose between the multiple routes. *Route 1* and *Route 2* are the two possible routes being chosen between.

Table 29. Route precedence			
Source comparison	Route 1 type	Route 2 type	Route chosen
Type 1	OSPF internal	RIP	OSPF internal
Type 1	OSPF internal	OSPF type 1 external	OSPF internal
Type 1	OSPF internal	OSPF type 2 external	OSPF internal
Type 1	RIP	OSPF type 1 external	Lowest cost route
Type 1	RIP	OSPF type 2 external	RIP route
Type 1	OSPF type 1 external	OSPF type 2 external	OSPF type 1 external
Type 2	OSPF internal	RIP	OSPF internal
Type 2	OSPF internal	OSPF type 1 external	OSPF internal
Type 2	OSPF internal	OSPF type 2 external	OSPF internal
Type 2	RIP	OSPF type 1 external	OSPF type 1 external

Table 29. Route precedence (continued)

Source comparison	Route 1 type	Route 2 type	Route chosen
Type 2	RIP	OSPF type 2 external	Lowest cost route
Type 2	OSPF type 1 external	OSPF type 2 external	OSPF type 1 external

OSPF configuration statements

This section contains descriptions of the following OSPF configuration statements:

- AREA
- AS_BOUNDARY_ROUTING
- COMPARISON
- DEMAND_CIRCUIT
- OSPF
- OSPF_INTERFACE
- RANGE
- ROUTERID
- VIRTUAL_LINK

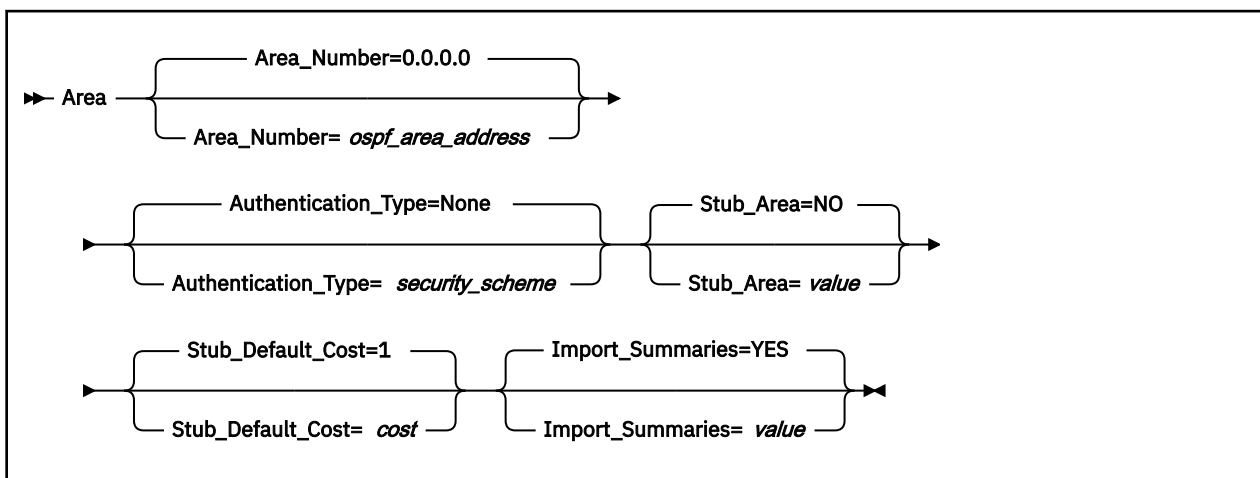
Use these statements to configure the OSPF environment for IPv4. For information about the statements used to configure IPv6 OSPF, see [“IPv6 OSPF configuration statements”](#) on page 241.

For information about how to display configuration information, see [“MSG Interface to the MPRoute Server”](#) on page 261.

AREA

Purpose

Sets the parameters for an OSPF area. If no areas are defined, MPRoute assumes that all directly attached networks belong to the backbone area (area ID 0.0.0.0).



Operands

Area_Number

The OSPF area number in dotted decimal.

Authentication_Type

The default security scheme to be used in the area. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. The area's default security scheme can be overridden on an interface basis by specifying the Authentication_Type keyword on OSPF_INTERFACE or VIRTUAL_LINK statements.

Stub_Area

Specifies whether this area is a stub area or not. Valid values are YES or NO.

If you specify Stub_area = YES, the area does not receive any AS external link advertisements, reducing the size of your database and decreasing memory usage for routers in the stub area. You cannot configure virtual links through a stub area. You cannot configure a router within the stub area as an AS boundary router.

You cannot configure the backbone as a stub area. External routing in stub areas is based on a default route. Each border area router attaching to a stub area originates a default route for this purpose. The cost of this default route is also configurable with the AREA statement.

Stub_Default_Cost

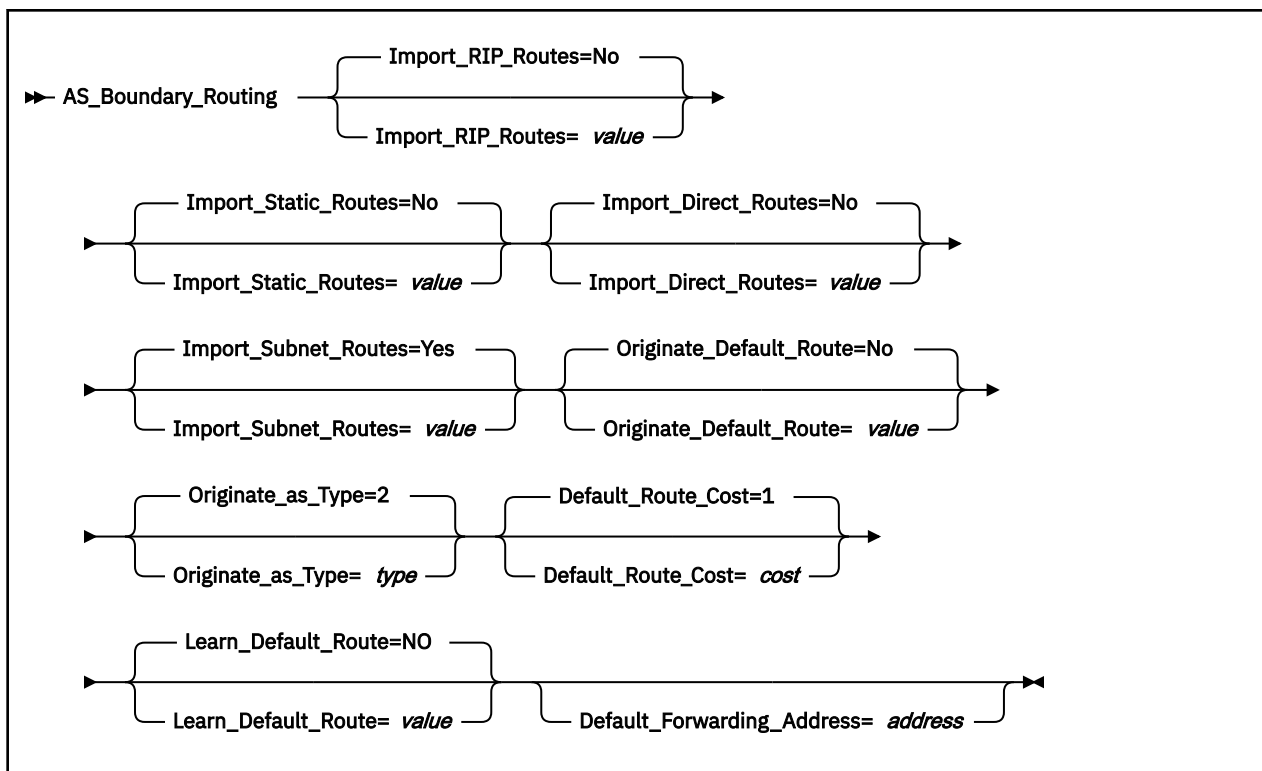
The cost that an MPRoute area border router associates with the default route that it generates into the stub area. Valid values are 1 to 16777215.

Import_Summaries

Determines whether this stub area will import a routing summary from a neighbor area. Valid values are YES or NO.

AS_BOUNDARY_ROUTING**Purpose**

Enables the AS boundary routing capability, which allows you to import routes learned from other methods (RIP, statically configured, and direct routes) into the OSPF domain. This statement must be coded even if the only route you want to import is the default route (destination 0.0.0.0). All routes are imported as either Type 1 or Type 2 external routes, depending on what was coded on the Comparison statement. The metric type used when importing routes determines how the imported cost is viewed by the OSPF domain. When comparing Type 2 metrics, only the external cost is considered in picking the best route. When comparing Type 1 metrics, the external and internal costs of the route are combined before making the comparison.



Operands

Import_RIP_Routes

Specifies whether routes learned by RIP will be imported into the OSPF routing domain. Valid values are YES or NO.

Import_Static_Routes

Specifies whether static routes (routes defined to the TCP/IP stack using the GATEWAY statement) will be imported into the OSPF routing domain. Valid values are YES or NO.

Import_Direct_Routes

Specifies whether direct routes will be imported into the OSPF routing domain. Valid values are YES or NO.

Import_Subnet_Routes

Independent of the RIP, static, and direct routes you may choose to import, you can also configure whether or not to import subnet routes into the OSPF domain. Valid values are YES or NO.

Originate_Default_Route

Specifies whether or not this router will originate an AS External default route into the OSPF domain. If YES and Default_Forwarding_Address is not also coded (or is coded to 0.0.0.0), this router will advertise itself as a default router. Valid values are YES or NO.

Originate_as_Type

Specifies the external type assigned to the default route. Valid values are 1 or 2.

Default_Route_Cost

Specifies the cost that OSPF associates with the default route. Valid values are 0 to 16 777 215.

Learn_Default_Route

Specifies whether OSPF will learn default routes from inbound RIP or OSPF external packets when their cost is equal to or higher than the default route originated by this host. Valid values are YES or NO.

Default_Forwarding_Address

If Originate_Default_Route is YES, this optional parameter may be used to specify that this router should originate a default route on behalf of a different router. This parameter is not needed if this router is to advertise itself as the default router. It should only be used when the default router is

another router that this router can route to, which is not capable of advertising an OSPF default route on its own behalf. In that case, this parameter should be set to a reachable interface IP address on the other router.

Restriction: This address must be reachable using an OSPF intra-area or inter-area route (labelled as SPF or SPIA in the RTTABLE display, or labelled as DIR but using an OSPF interface). This route could be a host, subnet, network, or default route. If no eligible route is found, the forwarding address is not included in the advertisements generated by this statement.

COMPARISON

Purpose

Use the COMPARISON statement as an alternate method for specifying the Comparison parameter on the OSPF configuration statement. See [“OSPF” on page 221](#) for a description of this statement.

For additional information about the COMPARISON configuration statement, see [“Notes on defining route precedence in a multiprotocol environment” on page 216](#).



Operands

Comparison

Compare to type 1 or 2 externals. Valid values are Type1 (or 1) or Type2 (or 2).

DEMAND_CIRCUIT

Purpose

Use the DEMAND_CIRCUIT statement as an alternate method for specifying the DEMAND_CIRCUIT parameter on the OSPF configuration statement. See [“OSPF” on page 221](#) for a description of this statement.



Operands

Demand_Circuit

Valid values are YES or NO.

OSPF

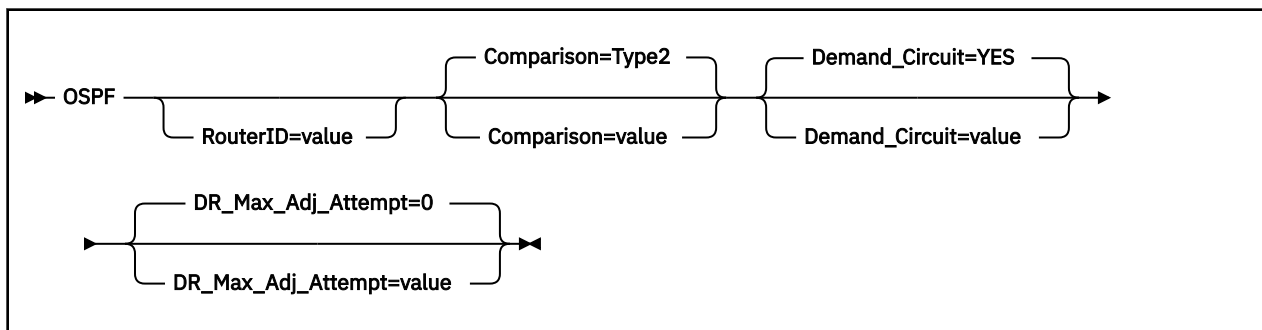
Purpose

Use the OSPF statement to specify various parameters that apply globally to IPv4 OSPF, either to all interfaces or to the overall OSPF autonomous system.

This statement is intended to replace the following standalone statements:

- ROUTERID
- COMPARISON
- DEMAND_CIRCUIT

Those standalone statements will continue to be supported. However, the OSPF statement is the preferred method for defining them, and future potential standalone parameters will be added to this statement only. If both the OSPF statement and the standalone statements are coded, the last one coded in the configuration file takes precedence.



Operands

RouterID

Every router in an IPv4 OSPF routing domain must be assigned a unique 32-bit router ID.

The value used for the OSPF router ID is chosen as follows:

- If this RouterID parameter is specified, the value configured is used as the OSPF router ID. This value must be one of the stack's configured OSPF interface IP addresses.

Rule: Loopback addresses are not valid IP interface addresses.

- If the RouterID is not configured, one of the OSPF interface addresses will be used as the OSPF router ID.

Valid values are any IPv4 dotted-decimal address that matches a configured OSPF interface.

Comparison

Tells MPRoute where external routes fit in the IPv4 OSPF hierarchy. OSPF supports two types of external metrics. Type 1 external metrics are equivalent to the link state metric. Type 2 external metrics are greater than the cost of any path internal to the autonomous system. Use of type 2 external metrics assumes that routing between autonomous systems is the major cost of routing a packet, and eliminates the need for conversion of external costs to internal link state metrics. For more information on the COMPARISON configuration parameter, see [“Notes on defining route precedence in a multiprotocol environment”](#) on page 216. Valid values are Type1 (or 1) or Type2 (or 2).

Demand_Circuit

Global demand circuit setting. Coding YES enables demand circuits. Demand circuit parameters can then be coded on the OSPF_Interface statement. Valid values are Yes or No.

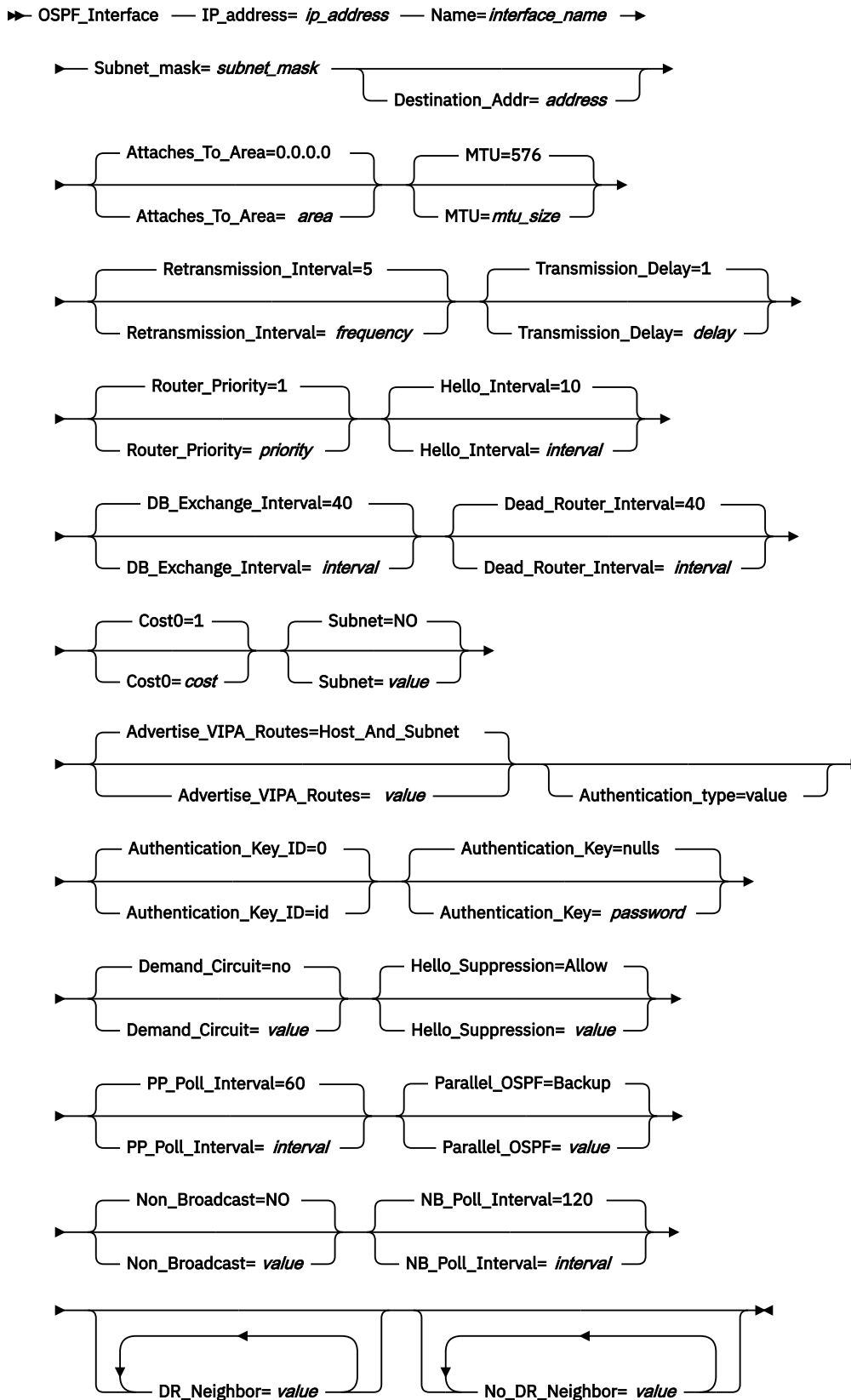
DR_Max_Adj_Attempt

Specifies the maximum number of adjacency attempts to be used for reporting and controlling futile neighbor state loops. After the adjacency attempt count for a neighboring designated router reaches the threshold, an informational message is issued to report the problem. If a redundant interface that can reach the neighbor is available, adjacency formation is attempted over that interface. An informational message is issued to report the interface switch and adjacency formation attempt. Valid values are in the range 0–100. The value 0 specifies infinite retries. For more information, see [“Preventing futile neighbor state loops during adjacency formation”](#) on page 198.

OSPF_INTERFACE

Purpose

Sets the OSPF parameters for interfaces. This statement will need to be replicated in the configuration file for each IP interface over which OSPF will operate.



Operands

IP_address

IP address of the local interface to be configured for OSPF.

The IP address can be a valid IP address that is configured on the system, or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are below. The result of coding a wildcard value is that all configured interfaces whose IP address matches the wildcard will be configured as OSPF interfaces. Configured interface IP addresses and names will be matched against possible wildcards in the order they appear below with the name and any matching wildcard being the best match, x.y.z.* being second best, and so forth.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Name

The name of the interface. The name must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP configuration file. Valid values are any strings from 1–16 characters in length, except numeric interface names containing a decimal point (for example 123.456). If an exact IP address match is not found, then this parameter is used first when searching wildcard addresses. If a definition matching name and any matching wildcard are not found, then the most specific wildcard address that matches is used. The same wildcard address can be configured more than once with unique names.

Subnet_Mask

The subnet mask of the subnet to which this interface attaches. This value must be the same for all routers attached to a common network. For more information, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Destination_Addr

IP address of the host at the remote end of this interface. This parameter is only valid for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface will not be added to the TCP/IP route table until OSPF communication is established with that host. A subnet route for the interface will be added at MPRoute initialization independent of whether this parameter is specified.

Attaches_To_Area

OSPF area to which this interface attaches. Valid values are 0.0.0.0 (the backbone), or any area defined by the AREA statement.

MTU=size

Specifies the MTU size for OSPF to add to the routing table for routes that take this interface. The size must be 0, or a number in the range of 576 to 65535.

If the LINK statement in the TCP/IP configuration file associated with *interface_name* has an MTU size (MTU *mtusize*), specify 0; a non-zero value is ignored.

Retransmission_Interval

Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from 1 to 65535 seconds.

If this parameter is set too low, needless retransmissions will occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

Transmission_Delay

This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the interface. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state

advertisement is sent out from this interface, it will be aged by this configured transmission delay. Valid values are 1 to 65535 seconds.

Router_Priority

This value is used for broadcast and nonbroadcast multiaccess networks to elect the designated router, with the highest priority router being elected. Valid values are 0 to 255.

A value of 0 indicates that MPSRoute will never become the designated router. A value of 1 indicates the lowest possible eligible priority and a value of 255 indicates the highest possible priority.

Hello_Interval

This parameter defines the number of seconds between OSPF Hello packets being sent out on this interface. This value must be the same for all routers attached to a common network. Valid values are 1 to 255 seconds.

DB_Exchange_Interval

The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure will be restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval will be set to the Dead_Router_Interval. Valid values are 2 through 65535.

Dead_Router_Interval

The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Setting this value too close to the Hello_Interval can result in the collapse of adjacencies. A value of 4*Hello_Interval is recommended. This value must be the same for all routers attached to a common network. Valid values are 2 to 65535.

Cost0

The OSPF cost for this interface. The cost is used to determine the shortest path to a destination. Valid values are 1 to 65535.

Subnet

The meaning of this parameter depends on the interface type.

For an interface to a point-to-point link, this option enables the advertisement of a stub route to the subnet that represents the link rather than the host route for the other router's address. In effect, this parameter controls whether, for this interface, MPSRoute implements option 1 (SUBNET=NO) or option 2 (SUBNET=YES) described in RFC 2328 (OSPF version 2) section 12.4.1.1.

For a VIPA interface, this option suppresses advertisement of either the VIPA host or subnet route. Normally TCP/IP advertises both a host route and a subnet route for owned VIPA interfaces. With this option set to NOVIPAHOST, the VIPA host route will be suppressed and only the VIPA subnet route will be advertised. With this option set to NOVIPASUBNET, the VIPA subnet route will be suppressed and only the VIPA host route will be advertised.

Legal values are:

- YES
- NO
- NOVIPASUBNET
- NOVIPAHOST

Note: NOVIPAHOST has the same effect as SUBNET=YES or ADVERTISE_VIPA_ROUTES=SUBNET_ONLY, and NOVIPASUBNET is equivalent to setting ADVERTISE_VIPA_ROUTES=HOST_ONLY

Guideline: The ADVERTISE_VIPA_ROUTES option is the preferred method to suppress VIPA advertisements.

Rules:

- Do not use this option for any VIPA whose subnet might exist on multiple hosts. If you do, problems can occur routing to all VIPAs that share the subnet.
- Do not specify SUBNET=NOVIPAHOST (or SUBNET=YES) for any VIPA whose subnet might exist on multiple hosts. If you do, problems can occur routing to all VIPAs that share the subnet.

Tips:

- Specifying SUBNET=YES on a VIPA interface has the same effect as specifying SUBNET=NOVIPAHOST.
- In order to fully suppress the VIPA subnet route, SUBNET=NOVIPASUBNET must be specified on every VIPA OSPF_INTERFACE statement that defines a VIPA in a common subnet.

Advertise_VIPA_Routes

This option is only valid on VIPA interfaces and controls how MPRoute will advertise the VIPA address. The default value of HOST_AND_SUBNET advertises both the VIPA host and subnet route. With this option set to HOST_ONLY, only the VIPA host route will be advertised. With this option set to SUBNET_ONLY, only the VIPA subnet route is advertised.

The value specified on the ADVERTISE_VIPA_ROUTES option will override any value specified on the SUBNET option. Legal values are:

- HOST_AND_SUBNET
- HOST_ONLY
- SUBNET_ONLY

Rule: Do not specify SUBNET_ONLY for any VIPA whose subnet might exist on multiple hosts. Problems can occur routing to all VIPAs that share the subnet when the subnet exists on multiple hosts.

Tip: The HOST_ONLY option must be specified for every VIPA in a common subnet. If the HOST_ONLY option is not specified for every VIPA in a common subnet, MPRoute will still advertise the VIPA subnet route for the interfaces not specifying HOST_ONLY.

Authentication_Type

The security scheme to be used on the network to which the interface attaches. If parameter is not specified, takes on the default value specified for the area to which the interface is attached. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. All hosts on the network must be configured with the same security scheme.

Authentication_Key_ID

The identifier of the authentication key defined with the AUTHENTICATION_KEY keyword. This is a constant numeric value from 0 to 255, with a default value of 0. It is only relevant when MD5 cryptographic authentication is employed on the interface; otherwise, it is ignored. This field is provided for compatibility with other routers that might require identification of a key identifier with the authentication key.

Authentication_Key

The value of the authentication key for this interface. This value must be the same for all routers attached to a common medium. The coding of this parameter depends on the authentication type being used on this interface.

For authentication type *none*, this parameter is not required and is ignored if coded.

For authentication type *password*, code the password for OSPF routers that are attached to this subnet. Valid values are any characters from EBCDIC code page 1047 up to 8 characters in length coded within double quotation marks or any hexadecimal string up to 8 bytes (16 hexadecimal characters) long that begins with 0x.

For authentication type *MD5*, code the 16-byte MD5 authentication key for OSPF routers attached to this subnet. This value can be coded in one of the following ways:

- The standard method is with a 16-byte hexadecimal string beginning with 0x (0x plus 32 hexadecimal characters).
- An additional method, which provides compatibility with Cisco, Extreme, and other vendor routers that use a Cisco-compatible CLI interface is to code the MD5 key as an ASCII string, specified

in double quotation marks prefixed with A. For example, to be compatible with this Cisco key definition, code the following:

```
ip ospf message-digest-key 4 md5 ABCDEFGHIJKLMNOP
```

This value would be coded in MPRoute as follows:

```
AUTHENTICATION_KEY_ID =4
AUTHENTICATION_KEY = A"ABCDEFGHIJKLMNOP"
```

Demand_Circuit

This parameter, when coded with YES, causes Link State Advertisements (LSAs) to not be periodically refreshed over this interface. Only LSAs with real changes will be advertised. In addition, coding this parameter to YES causes LSAs flooded over this interface to never age out. Valid values are YES or NO.

Hello_Suppression

This parameter is only used on point-to-point interfaces that are demand circuits. It allows you to configure the interface for Hello Suppression. Valid values are ALLOW, REQUEST, or DISABLE.

If either or both sides specify DISABLE, Hello_Suppression is disabled. If both specify ALLOW, Hello_Suppression is disabled. If one specifies ALLOW and the other REQUEST, or if both specify REQUEST, Hello_Suppression is enabled.

PP_Poll_Interval

This parameter specifies the interval (in seconds) that MPRoute should use when attempting to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available. This parameter is meaningful only if Demand_Circuit is coded YES and Hello_Suppression has been enabled. Valid values are 0 to 65535.

Parallel_OSPF

This parameter designates whether the OSPF interface is primary or backup when more than one OSPF interface is defined to the same subnet. Only one of these interfaces can be configured as primary, meaning that it will be the interface to carry the OSPF protocol traffic between MPRoute and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If the primary interface is later reactivated, OSPF traffic will not be automatically switched back from the backup interface to the primary interface. If you want to switch OSPF traffic back to the primary interface, the backup interface must be stopped. If none of the interfaces to the common subnet are configured as primary, a primary interface will be selected by MPRoute. Valid values are BACKUP and PRIMARY.

Non_Broadcast

If the router is connected to a nonbroadcast, multiaccess network (NBMA), such as Frame Relay or Hyperchannel networks, coding a Non_Broadcast helps the router discover its neighbors. This can also be coded for a broadcast-capable network when you want MPRoute to unicast its packets instead of multicasting them. In addition to coding this parameter, each neighbor must be configured with the DR_NEIGHBOR parameter, for those neighbors that are eligible to become the designated router, or NO_DR_NEIGHBOR for those neighbors that are not eligible to become the designated router. This statement is ignored when this OSPF interface is coded as a wildcard. Valid values are YES or NO.

NB_Poll_Interval

This parameter specifies the frequency (in seconds) of hellos sent to neighbors that are inactive. You must set this poll interval consistently across all interfaces that attach to the same subnetwork for OSPF to function correctly. This statement is only valid when Non_Broadcast is coded as YES. Valid values are 1 to 65535.

DR_Neighbor

Configures designated router eligible neighbors adjacent to the router over this interface. In nonbroadcast multiaccess networks, neighbors need to be configured to all OSPF routers on the network. Multiple DR_Neighbor statements may be coded on an OSPF_interface statement as necessary.

Note: It is not necessary or recommended to define neighbors on broadcast- or multicast-capable media. If you do define neighbors on these media, MPRoute will be able to communicate OSPF

information *only* with those neighbors that are defined (it will not form adjacencies with any additional neighbors).

No_DR_Neighbor

Configures non-designated router eligible neighbors adjacent to the router over this interface. In nonbroadcast multiaccess networks, neighbors need to be configured to all OSPF routers on the network. Multiple No_DR_Neighbor statements may be coded on an OSPF_Interface statement as necessary.

Note: It is not necessary or recommended to define neighbors on broadcast or multicast capable media. If you do define neighbors on these media, MPRoute will be able to communicate OSPF information only with those neighbors that are defined (it will not form adjacencies with any additional neighbors).

Usage Notes

When configuring multiaccess parallel interfaces (primary and secondary interfaces having IP addresses in the same network) for MPRoute(OSPF), the parallel interfaces order in the HOME list of TCP/IP configuration file must match the order of the corresponding OSPF_INTERFACE statements in the MPRoute configuration file. This causes MPRoute to treat the first interface in the list as primary and the remaining ones as secondary. The interfaces order is critical for MPRoute(OSPF) to be able to send the LSAs correctly to the neighboring routers. This allows the primary interface to be recognized. Otherwise, a secondary interface configured in MPRoute or HOME list might be inadvertently treated as a primary interface, and this can cause routing problems between MPRoute and its neighbors. In case of a primary interface failure, MPRoute will use the first available secondary interface and mark it as primary.

RANGE

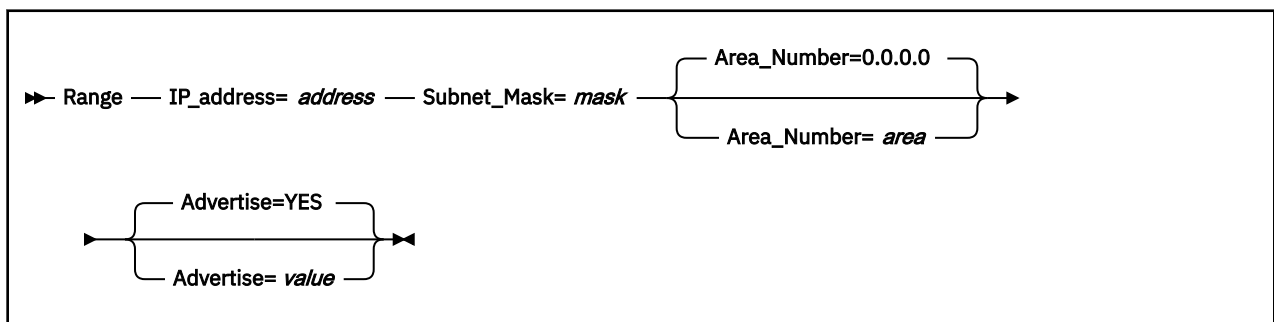
Purpose

Adds ranges to OSPF areas. OSPF areas can be defined in terms of address ranges. External to the area, a single route is advertised for each address range. For example, if an OSPF area were to consist of all subnets of the class B network 128.185.0.0, it would be defined as consisting of a single address range. The address range would be specified as an address of 128.185.0.0 together with a mask of 255.255.0.0. Outside of the area, the entire subnetted network would be advertised as a single route to network 128.185.0.0.

Ranges can be defined to control which routes are advertised external to an area.

When OSPF is configured not to advertise the range, no interarea routes are advertised for routes that fall within the range. Ranges cannot be used for areas that serve as transit areas for virtual links.

Note: This will not prevent AS-external routes from being advertised if used in conjunction with the AS_BOUNDARY statement.



Operands

IP_Address

Common subnet portion of IP addresses in this range. Valid values are valid network and subnetwork addresses.

Subnet_Mask

Subnet mask with respect to the network range defined in IP_Address.

Area_Number

Area number for which to add this range. Valid values are any defined areas.

Advertise

Specifies whether this range will be advertised to other areas. Valid values are YES or NO.

RouterID

Purpose

Use the RouterID statement as an alternate method for specifying the RouterID parameter on the OSPF configuration statement. See [“OSPF” on page 221](#) for a description of this statement.

- If the RouterID statement is specified, the value configured is used as the OSPF router ID. This value must be one of the stack's configured OSPF interface IP addresses.

Rule: Loopback addresses are not valid OSPF interface IP addresses.

- If the RouterID is not configured, one of the OSPF interface addresses will be used as the OSPF router ID.

```
➤ RouterID= id ➤
```

Operands

RouterID

A dotted-decimal value as previously described.

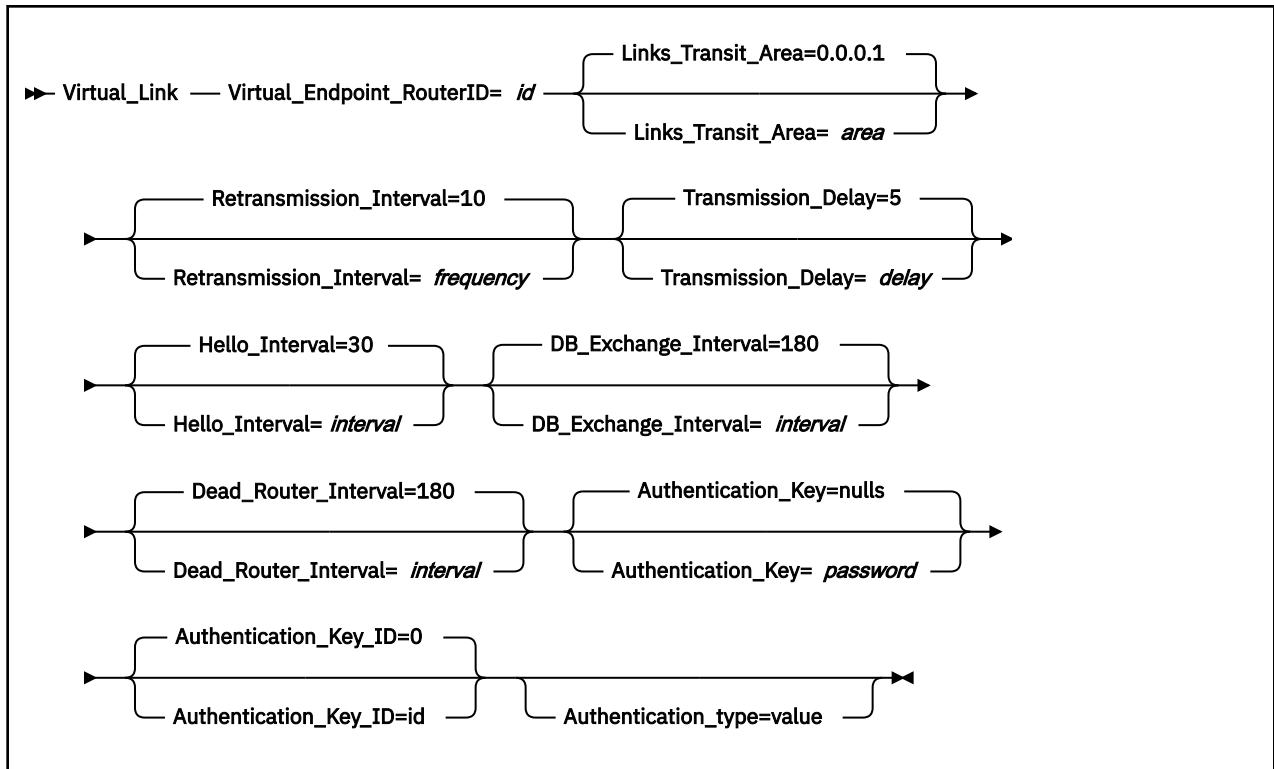
VIRTUAL_LINK

Purpose

Configures a virtual link between two area border routers. To maintain backbone connectivity you must have all of your backbone routers interconnected either by permanent or virtual links. Virtual links are considered to be separate router interfaces connecting to the backbone area. Therefore, you are asked to specify many of the interface parameters when configuring a virtual link.

Virtual links can be configured between any two backbone routers that have an interface to a common nonbackbone, nonstub area. Virtual links are used to maintain backbone connectivity and must be configured at both endpoints.

Tip: OSPF virtual links are not to be confused with Virtual IP Address support (VIPA).



Operands

Virtual_Endpoint_RouterID

Router ID of the virtual neighbor (other endpoint). Router IDs are entered in the same form as IP addresses.

Links_Transit_Area

This is the nonbackbone, nonstub area through which the virtual link is configured. Virtual links can be configured between any two area border routers that have an interface to a common nonbackbone and nonstub area. Virtual links must be configured in each of the link's two endpoints. Valid values are any area defined by the AREA statement, except 0.0.0.0.

Retransmission_Interval

Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from 1 to 65535 seconds.

If this parameter is set too low, needless retransmissions will occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

Transmission_Delay

This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the virtual link. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent out from this virtual link, it will be aged by this configured transmission delay. Valid values are 1 to 65535 seconds.

Hello_Interval

This parameter defines the number of seconds between OSPF Hello packets being sent out from this virtual link. Valid values are 1 to 255 seconds. The Hello_Interval should be set higher than the same value used on the intervening, actual OSPF interfaces.

DB_Exchange_Interval

The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure will be restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval will be set to the Dead_Router_Interval. Valid values are 2 through 65535.

Dead_Router_Interval

The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Valid values are 2 to 65535. The dead router interval should be set higher than the same value used on the intervening, actual, OSPF interfaces.

Authentication_Key

The value of the authentication key for this interface. This value must be the same for all routers attached to a common medium. The coding of this parameter depends on the authentication type being used on this interface.

For authentication type *none*, this parameter is not required and is ignored if coded.

For authentication type *password*, code the password for OSPF routers that are attached to this subnet. Valid values are any characters from EBCDIC code page 1047 up to 8 characters in length coded within double quotation marks or any hexadecimal string up to 8 bytes (16 hex characters) long that begins with 0x.

For authentication type *MD5*, code the 16-byte MD5 authentication key for OSPF routers attached to this subnet. This value can be coded in one of the following ways:

- The standard method is with a 16-byte hexadecimal string beginning with 0x (0x plus 32 hexadecimal characters).
- An additional method, which provides compatibility with Cisco, Extreme, and other vendor routers that use a Cisco-compatible CLI interface is to code the MD5 key as an ASCII string, specified in double quotation marks prefixed with A. For example, to be compatible with this Cisco key definition code the following:

```
ip ospf message-digest-key 4 md5 ABCDEFGHIJKLMNOP
```

This value would be coded in MPRoute as follows:

```
AUTHENTICATION_KEY_ID =4
AUTHENTICATION_KEY = A"ABCDEFGHIJKLMNPO"
```

Authentication_Key_ID

The identifier of the authentication key defined with the AUTHENTICATION_KEY keyword. This is a constant numeric value from 0-255, with a default value of 0. It is only relevant when MD5 cryptographic authentication is employed on the virtual link; otherwise, it is ignored. This field is provided for compatibility with other routers which might require identification of a key identifier with the authentication key.

Authentication_Type

The security scheme to be used over the virtual link. If not specified, the statement takes on the default value specified for the backbone area. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. Both hosts attached to the virtual link must be configured with the same security scheme.

RIP configuration statements

This section contains descriptions of the following RIP configuration statements.

- ACCEPT_RIP_ROUTE
- FILTER
- IGNORE_RIP_NEIGHBOR
- ORIGINATE_RIP_DEFAULT
- RIP_INTERFACE
- SEND_ONLY

These statements are for configuring the RIP environment for IPv4. For information on the statements to be used for configuring IPv6 RIP, see [“IPv6 RIP protocol” on page 197](#).

ACCEPT_RIP_ROUTE

Purpose

Allows a network, subnet, or host route to be accepted independent of whether the interface it was received on has the corresponding reception parameter enabled (network, subnet, or host). Routes added in this manner can be thought of as a list of exception conditions.

Note: Coding this statement will not enable updates for this destination to be received on RIP interfaces with RECEIVE_RIP=NO coded. Also, this will not override RIP version filters coded using the RECEIVE_RIP parameter on RIP_INTERFACE statements. For example, on a RIP_INTERFACE with RECEIVE_RIP=RIP2, a RIPV1 route that would otherwise be allowed by this statement will not be received.

```
➤ Accept_RIP_Route — IP_address= address ➤
```

Operands

IP_address

Destination route to be unconditionally accepted.

FILTER

Purpose

The filter statement can be coded stand-alone in the MPRoute configuration file (nosend and noreceive only) to apply to all configured RIP interfaces.

```
➤ filter= (filter_type,dest_route,filter_mask) ➤
```

Operands

filter_type

The *filter_type* can be any of the following values:

nosend

Specifies that routes matching the *dest_route* and *filter_mask* are not to be broadcast over RIP interfaces. This option serves as an RIP output filter.

noreceive

Specifies that routes matching the *dest_route* and *filter_mask* are to be ignored in broadcasts received over RIP interfaces. This option serves as a RIP input filter.

dest_route

The *dest_route* specifies the destination route in network, subnetwork, or host format in dotted decimal form. Alternatively, an asterisk (*), which matches *any* destination, can be coded to filter out all routes sent or received over an interface. The use of the asterisk is also referred to as a blackhole filter. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: When the Originate_RIP_Default statement is configured, the blackhole nosend filter will not prevent sending of the default route.

filter_mask

The *filter_mask* specifies the filter mask in dotted decimal form. If not coded, the default filter mask will be 255.255.255.255, meaning apply the filter to the *dest_route* as coded. Coding the filter mask has no meaning and is not valid if the *dest_route* is coded as an asterisk (*) for a *blackhole* filter.

IGNORE_RIP_NEIGHBOR

Purpose

Specifies that RIP routing table broadcasts from this gateway are to be ignored. This option serves as an RIP input filter.



Operands

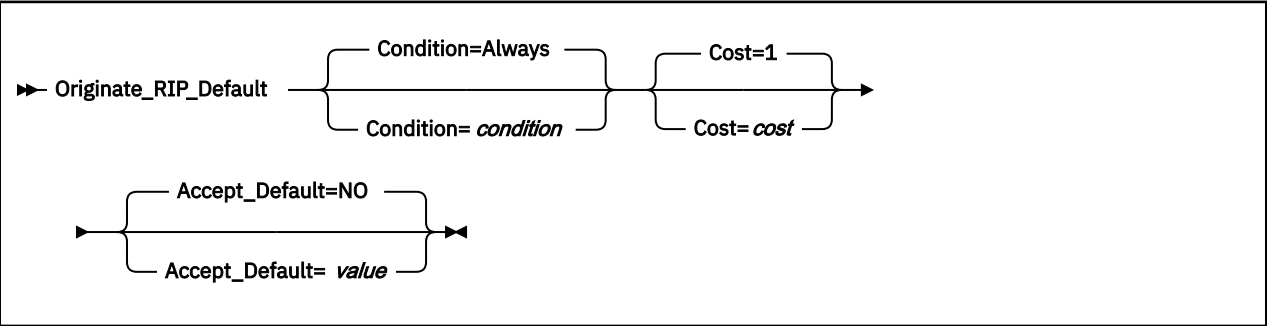
IP_address

Specifies the IP address of the gateway from which routing table broadcasts will be ignored. For multiple IP addresses, the statement must be repeated for each IP address.

ORIGINATE_RIP_DEFAULT

Purpose

Indicates under what conditions RIP will support Default route (destination/mask 0.0.0.0/0.0.0.0) generation.



Operands

Condition

Condition for when RIP is to advertise this router as a default router. Valid values are:

Always

Always originate RIP default.

OSPF

Originate RIP default if OSPF routes are available.

Never

Never advertise this router as a default router.

Cost

Specifies the cost that RIP will advertise with the default route that it originates. Valid values are 1 to 16.

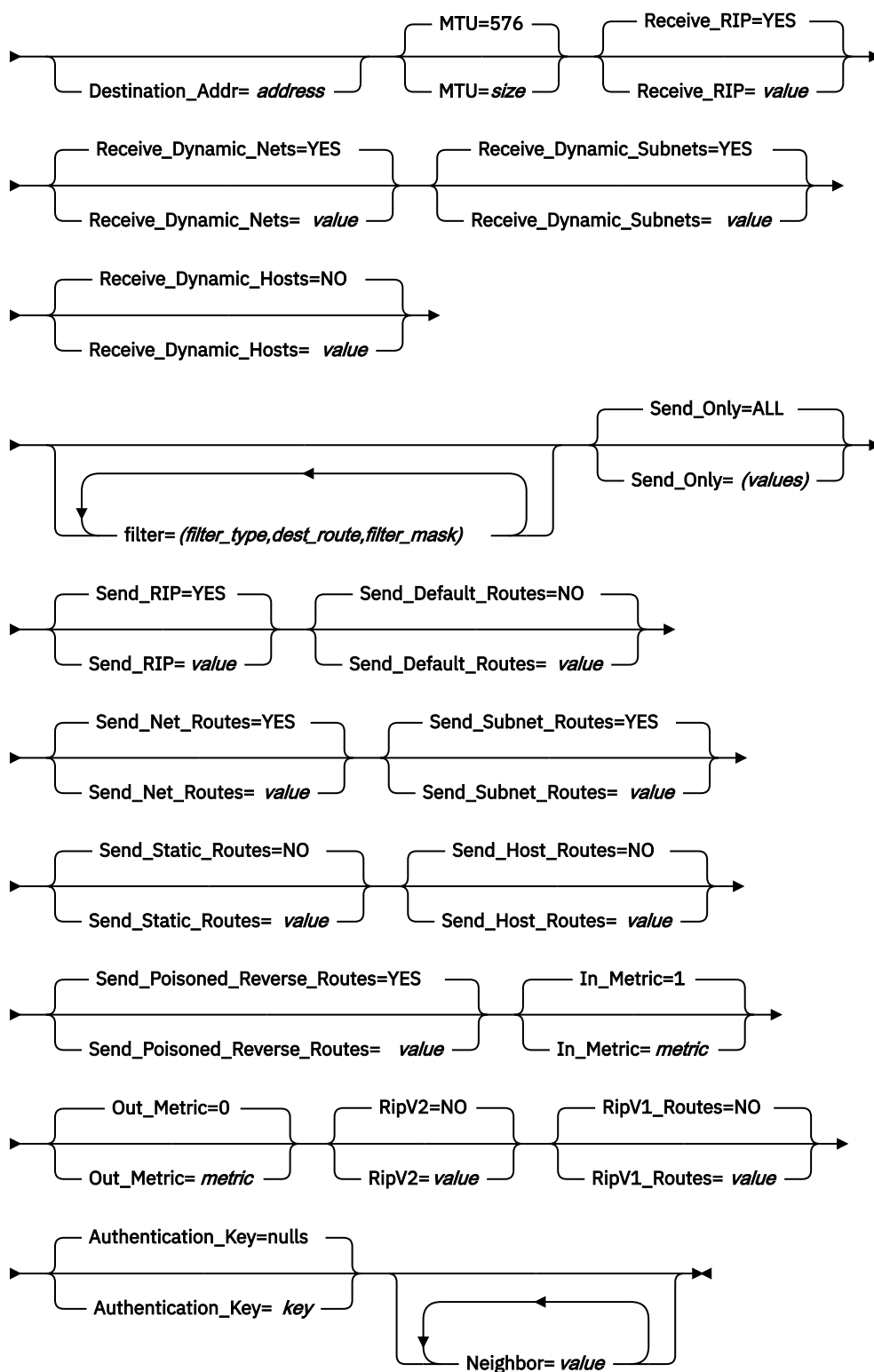
Accept_Default

If RIP learns of a default route that has a lower cost than is specified in this statement, that default route will be accepted and replace this router's default route. Additionally, if this parameter is coded to YES, then RIP will learn default routes from inbound RIP packets, even if their cost is higher than default routes originated by this host. Valid values are YES or NO.

RIP_INTERFACE**Purpose**

Configures the RIP parameters for each IP interface. This statement will need to be replicated in the configuration file for each IP interface over which RIP will operate.

►► RIP_Interface — IP_address= *address* — Name= *interface_name* — Subnet_mask= *subnet_mask* —►



Operands

IP_address

IP address of interface to be configured for RIP.

The IP address can be a valid IP address that is configured on the system or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are below. The result of coding a wildcard value are that all configured interfaces whose IP address matches the wildcard will be configured as RIP interfaces. Configured interface IP addresses and names will be matched against possible wildcards in the order they appear below with the name and any matching wildcard being the best match, x.y.z.* being second best, and so forth.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Name

The name of the interface. Must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP configuration file. Valid values are any string from 1–16 characters in length, except numeric interface names containing a decimal point (for example, 123.456). If an exact IP address match is not found, then this parameter is used first when searching wildcard addresses. If a definition with a matching name and any matching wildcard are not found, then the most specific wildcard address that matches is used. The same wildcard address can be configured more than once with unique names.

Subnet_Mask

Subnet mask for the associated interface IP address. For more information, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Destination_Addr

IP address of the host at the remote end of this interface. This parameter is only valid for point-to-point links and is a required parameter for point-to-point links that cannot receive RIP2 packets (see RECEIVE_RIP for more information on the level of RIP packets that an interface can receive). If this parameter is not specified for a point-to-point link that can receive RIP2 packets, a route to the host at the remote end of the interface will not be added to the TCP/IP route table until RIP communication is established with that host. A subnet route for the interface will be added at MPRoute initialization independent of whether this parameter is specified.

MTU=size

Specifies the MTU size for RIP to add to the routing table for routes that take this interface. The size must be 0, or a number in the range of 576 to 65535.

If the LINK statement in TCP/IP configuration file associated with *interface_name* has an MTU size (MTU *mtusize*), specify 0; a non-zero value is ignored.

Receive_RIP

Specifies what type of RIP updates will be accepted over this interface. Valid values are:

RIP1

Accept only RIP version 1 updates over this interface.

RIP2

Accept only RIP version 2 updates over this interface.

ANY

Accept RIP Version 1 and RIP Version 2 updates over this interface.

Note: If RIP2 authentication is required and this value is coded, unauthenticated RIP1 packets *will* be received over this interface. Also, if RIP2 authentication is not required, authenticated RIP2 packets will *not* be received over this interface, regardless of the value of RIPV2.

YES

If RIPV2=YES, then receive only RIP Version 2 updates over this interface. If RIPV2=No, then receive only RIP Version 1 updates over this interface. This is the default value.

NO

No RIP packets will be received over this interface, regardless of any other filters.

Receive_Dynamic_Nets

Specifies whether or not to learn routes for networks over this interface. If this is not set, only nets explicitly allowed using the `Accept_RIP_Route` configuration statement will be accepted on this interface. Valid values are YES or NO.

Receive_Dynamic_Subnets

Specifies whether or not to learn routes for subnets over this interface. If this is not set, only subnets explicitly allowed using the `Accept_RIP_Route` configuration statement will be accepted on this interface. Valid values are YES or NO.

Receive_Dynamic_Hosts

Specifies whether or not to learn routes for hosts over this interface. If this is not set, only hosts explicitly allowed using the `Accept_RIP_Route` configuration statement will be accepted on this interface. Valid values are YES or NO.

filter

Multiple filter parameters can be coded on a `RIP_Interface` statement. When specified on the `RIP_Interface` statement, the filter parameter applies only to the corresponding RIP interface. The filter statement can also be coded stand-alone in the MPRoute configuration file (nosend and noreceive only) to apply to all configured RIP interfaces.

The *filter_type* can be any of the following values:

Value**Description****nosend**

Specifies that routes matching the `dest_route` and `filter_mask` are not to be broadcast over this interface. This option serves as an RIP output filter.

noreceive

Specifies that routes matching the `dest_route` and `filter_mask` are to be ignored in broadcasts received over this interface. This option serves as an RIP input filter.

send

Specifies that routes matching the `dest_route` and `filter_mask` are to be broadcast over only this interface (or any other RIP interface with an equivalent filter). This option serves as an RIP output filter and can be used for inbound and outbound traffic splitting.

send_cond

Specifies that routes matching the `dest_route` and `filter_mask` are to be broadcast over only this interface when this interface is active (or any other active RIP interface with an equivalent filter). If this interface is inactive, the routes can be broadcast over other interfaces. This option serves as an RIP output filter and can be used for inbound and outbound traffic splitting.

receive

Specifies that routes matching the `dest_route` and `filter_mask` are to be received over only this interface (or any other RIP interface with an equivalent filter). If received over other RIP interfaces, the routes are discarded. This option serves as an RIP input filter.

receive_cond

Specifies that routes matching the `dest_route` and `filter_mask` are to be received over only this interface when this interface is active (or any other active RIP interface with an equivalent filter). If this interface is inactive, the routes can be received over all other active RIP interfaces. This option serves as an RIP input filter.

The *dest_route* specifies the destination route in network, subnetwork, or host format in dotted decimal form. Alternatively, an asterisk (*) can be coded in conjunction with the nosend and noreceive filter types. This serves as a *blackhole* filter that can be used to filter out all routes broadcast or received over an interface. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: If the blackhole nosend filter is used, it will not filter out the sending of the default route when the Originate_RIP_Default statement is also configured.

The *filter_mask* specifies the filter mask in dotted decimal form. If not coded, the default filter mask will be 255.255.255.255, meaning apply the filter to the *dest route* as coded. Coding the filter mask has no meaning and is not valid if the *dest route* is coded as an asterisk (*) for a *blackhole* filter.

Send_Only

Specifies broadcast restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL

Specifies no broadcast restrictions.

VIRTUAL

Broadcasts virtual IP addresses.

DEFAULT

Broadcasts the default route.

DIRECT

Broadcasts direct routes.

TRIGGERED

Only broadcasts routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are Or'd together to determine what should be broadcasted. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) will broadcast virtual IP addresses and the default route. When ALL is coded it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the RIP_Interface statement, the Send_Only parameter applies only to the corresponding RIP interface. The Send_Only statement can also be coded stand-alone in the MPRoute configuration file to apply to all RIP interfaces.

Send_RIP

Specifies whether or not RIP advertisements will be broadcast over this interface. Valid values are YES or NO.

Send_Default_Routes

Advertise the default route (destination 0.0.0.0), if it is available, in RIP responses sent from this IP source address. Valid values are YES or NO.

Note: If DEFAULT is coded on the Send_Only parameter or the stand-alone Send_Only statement, the Send_Default_Routes parameter is ignored and will be set to YES.

Send_Net_Routes

Advertise all network level routes in RIP responses sent from this IP address. Valid values are YES or NO.

Send_Subnet_Routes

Advertise appropriate subnet-level routes in RIP responses sent from this IP address. Valid values are YES or NO.

In this context an appropriate subnet is one that meets RFC 1058 subnet advertisement constraints as follows:

- Natural Net must be the same as the IP source's natural net.
- Subnet mask must be the same.

Send_Static_Routes

Advertise static and direct routes in RIP responses sent from this IP source address. Split horizon is applied; that is, static routes configured over an interface will not be included in RIP responses sent from that interface. Valid values are YES or NO.

Send_Host_Routes

Advertise host routes in RIP responses sent from this IP source address. In this context, a host route is one with a mask of 255.255.255.255. Valid values are YES or NO.

Send_Poisoned_Reverse_Routes

Advertise poisoned reverse routes over the interface corresponding to the next hop. A poison reverse route is one with an infinite metric (16). Valid values are YES or NO. If NO is specified, MPRoute still uses split horizon.

In_Metric

Specifies the value of the metric to be added to RIP routes received over this interface prior to installation in the routing table. Valid values are 1 to 15.

Out_Metric

Specifies the value of the metric to be added to RIP routes advertised over this interface. Valid values are 0 to 15.

RipV2

Enables RIP V2 packets to be sent on this link. Valid values are YES or NO. If YES, all RIP packets sent on this link will be RIPV2. If NO, all RIP packets sent on this link will be RIPV1. See the RECEIVE_RIP description above for information about configuring the level of RIP packets that can be received on this link.

RipV1_Routes

Specifies whether RIP V1 routes should be advertised on this RIP V2 link. Valid values are YES or NO.

Authentication_Key

RIP V2 authentication key. Only used for RIP V2 packets. Coding this key will not prevent reception of unauthenticated RIP V1 packets. To ensure that only authenticated RIP packets can be received over this interface, code RECEIVE_RIP=RIP2 in addition to this parameter. Valid values are any alphanumeric string from code page 1047 up to 16 characters in length coded within double quotation marks, or any hexadecimal string which begins with 0x.

Rules:

- If the value is entered in characters (rather than the hexadecimal string), that value is case sensitive.
- If an authentication key is not provided, authenticated RIP V2 packets will not be received, even if RECEIVE_RIP=ANY.

Neighbor

Multiple Neighbor parameters can be coded on a RIP_Interface statement to indicate adjacent RIP routers. This should be used when the interface is not point-to-point, does not support broadcast, and does not support multicast. An examples of interface type for which the Neighbor parameter must be used is Hyperchannel.

Guideline: It is not necessary or recommended to define neighbors on multicast capable media if this interface supports RIP V2, or broadcast capable media for interfaces that support RIP V1 or RIP V2. If you do define neighbors on these media, MPRoute will be able to communicate RIP information *only* with those neighbors that are defined (it will not learn about any additional neighbors).

SEND_ONLY**Purpose**

The SEND_ONLY statement can be coded stand-alone in the MPRoute configuration file to apply to all RIP interfaces.



Operands

(values)

Specifies broadcast restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL

Specifies no broadcast restrictions.

VIRTUAL

Broadcasts virtual IP addresses.

DEFAULT

Broadcasts the default route.

DIRECT

Broadcasts direct routes.

TRIGGERED

Only broadcast routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are OR'd together to determine what should be broadcasted. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) will broadcast virtual IP addresses and the default route. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the SEND_ONLY statement in the MPRoute configuration file, it applies to all RIP_Interfaces. The SEND_ONLY parameter can also be coded on the RIP_INTERFACE statement. When specified on the RIP_INTERFACE statement, the SEND_ONLY parameter applies only to the corresponding RIP_Interface.

IPv6 OSPF configuration statements

This section contains descriptions of the following IPv6 OSPF configuration statements:

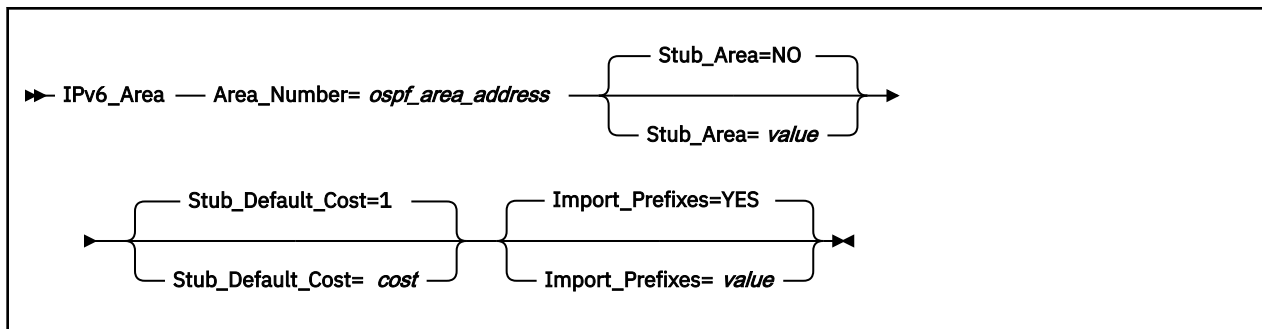
- IPv6_AREA
- IPv6_AS_BOUNDARY_ROUTING
- IPv6_OSPF
- IPv6_OSPF_INTERFACE
- IPv6_RANGE
- IPv6_VIRTUAL_LINK

For information about how to display configuration information, see [“MSG Interface to the MPRoute Server”](#) on page 261.

IPv6_AREA

Purpose

Sets the parameters for an IPv6 OSPF area. If no areas are defined, MPRoute assumes that all the router's directly attached networks belong to the backbone area (area ID 0.0.0.0).



Operands

Area_Number

The 32-bit OSPF area number in dotted decimal.

Stub_Area

Specifies whether this area is a stub area or not. Valid values are YES or NO.

Restrictions: If you specify Stub_area = YES, the area does not receive any AS external link advertisements, reducing the size of your database and decreasing memory usage for routers in the stub area. The following restrictions apply:

- You cannot configure virtual links through a stub area.
- You cannot configure a router within the stub area as an AS boundary router.
- You cannot configure the backbone as a stub area.

External routing in stub areas is based on a default route. Each area border router attaching to a stub area originates a default route for this purpose. The cost of this default route is also configurable with the IPv6_AREA statement.

Stub_Default_Cost

The cost that an MPRoute area border router associates with the default route that it generates into the stub area. Valid values are 1 to 16777215.

Import_Prefixes

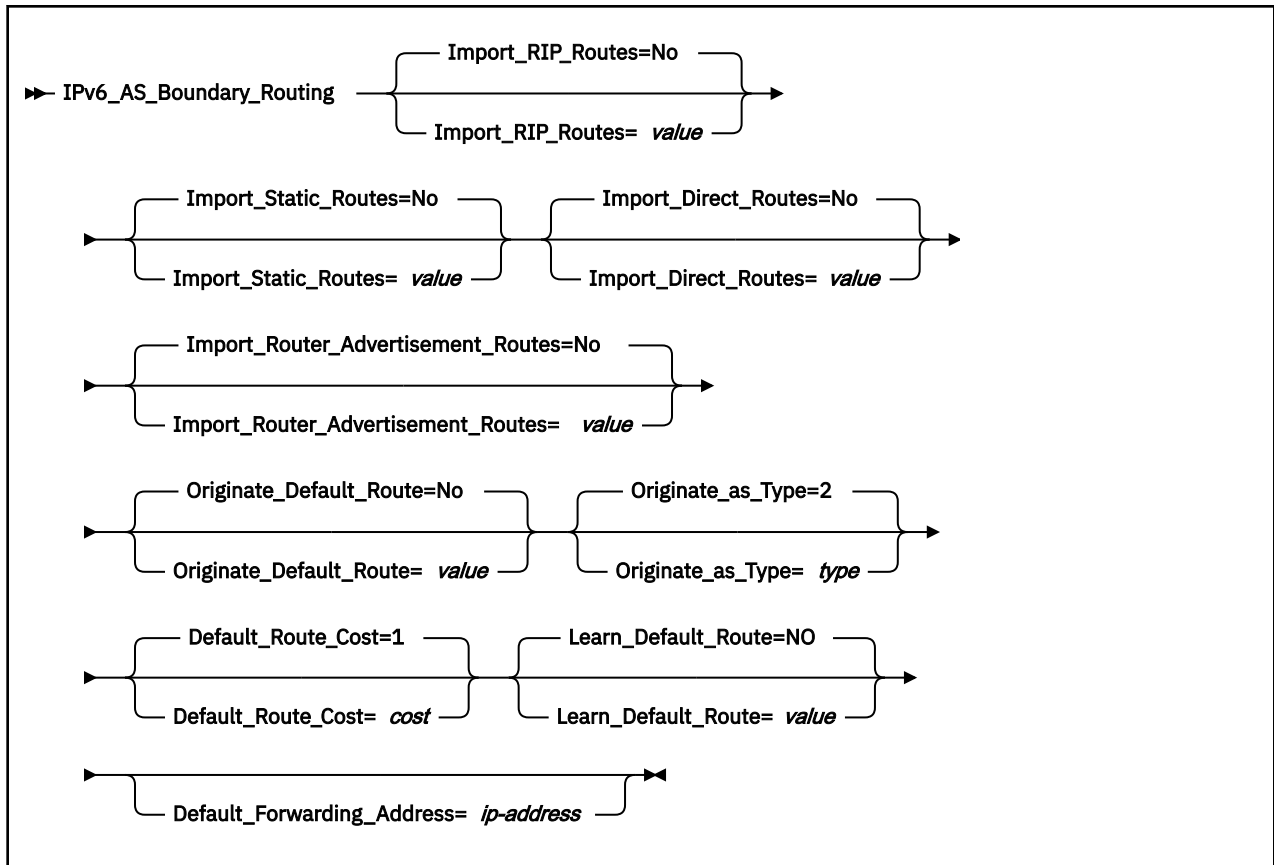
If this area is a stub area, indicates whether prefixes from neighboring areas will be imported. Valid values are YES or NO.

Tip: A stub area with Import_Prefixes set to NO is commonly referred to in RFCs and other standards documentation as a Totally Stubby Area.

IPv6_AS_BOUNDARY_ROUTING

Purpose

Enables the AS boundary routing capability, which allows you to import routes learned from other methods (IPv6 RIP, statically configured, or direct routes) into the IPv6 OSPF domain. This statement must be coded even if the only route you want to import is the default route (prefix length 0). All routes are imported as either Type 1 or Type 2 external routes, depending on what was coded on the Comparison statement. The metric type used when importing routes determines how the imported cost is viewed by the IPv6 OSPF domain. When comparing Type 2 metrics, only the external cost is considered in selecting the best route. When comparing Type 1 metrics, the external and internal costs of the route are combined before making the comparison.



Operands

Import_RIP_Routes

Specifies whether routes learned by IPv6 RIP will be imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

Import_Static_Routes

Specifies whether static routes (routes defined to the TCP/IP stack using the GATEWAY statement) will be imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

Import_Direct_Routes

Specifies whether IPv6 direct routes will be imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

Import_Router_Advertisement_Routes

Specifies whether routes learned by the TCP/IP stack from IPv6 Router Advertisements will be imported into the IPv6 OSPF routing domain. Valid values are YES and NO.

Tip: If a router is advertising a route into the OSPF domain on a link LSA it will be considered an OSPF internal route, regardless of whether or not it is also being advertised in an IPv6 Router Advertisement. Therefore, this parameter only controls routes that are only advertised by routers in IPv6 Router Advertisements.

Originate_Default_Route

Specifies whether or not this host will originate an AS External default route into the IPv6 OSPF domain. If YES and Default_Forwarding_Address is not also coded (or is coded to ::), this host will advertise itself as a default router. Valid values are YES or NO.

Originate_as_Type

Specifies the external type assigned to the default route originated by this host if Originate_Default_Route is YES. Valid values are 1 or 2.

Tip: See the comparison parameter in “IPv6_OSPF” on page 244 for more information on external route types.

Default_Route_Cost

Specifies the cost that IPv6 OSPF associates with the default route originated by this host if Originate_Default_Route is YES. Valid values are 0 to 16 777 215.

Learn_Default_Route

Specifies whether IPv6 OSPF will learn default routes from inbound packets when their cost is equal to or higher than the cost of the default route originated by this host. Valid values are YES or NO. If this parameter is set to NO, then only default routes with lower cost than the one originated by this host will be learned.

Default_Forwarding_Address

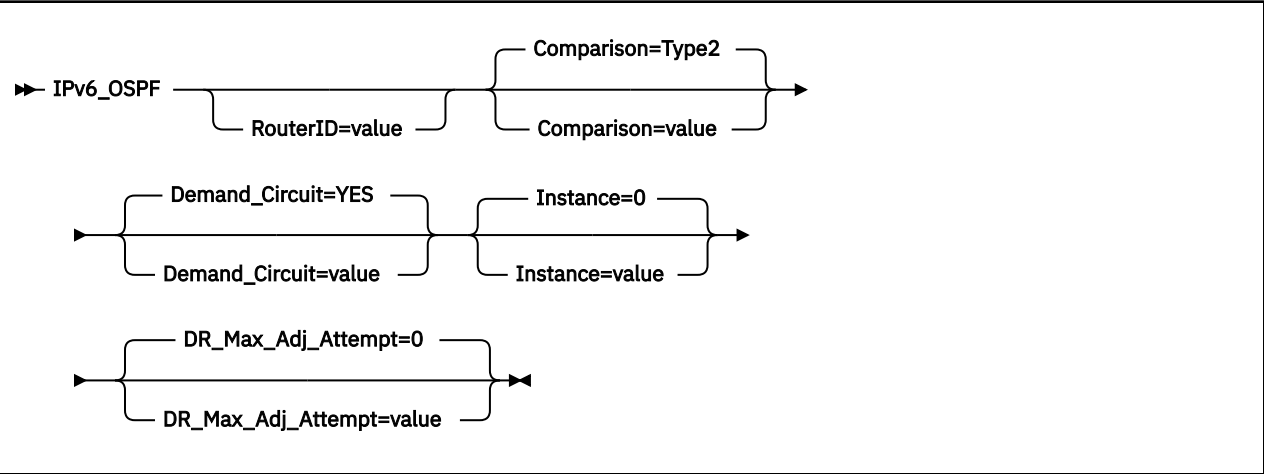
If Originate_Default_Route is YES, this optional parameter may be used to specify that this host should originate a default route on behalf of a different router. This parameter is not needed if this host is to advertise itself as the default router. It should only be used when the default router is another router that this host can route to, which is not capable of advertising an IPv6 OSPF default route on its own behalf. In that case, this parameter should be set to a reachable IP address on the other router.

Restriction: This address must be reachable using an OSPF intra-area or inter-area route (labelled as SPF or SPIA in the RT6TABLE display, or labelled as DIR but using an OSPF interface). This route could be a host, prefix, or default route. If no eligible route is found, the forwarding address is not included in the advertisements generated by this statement.

IPv6_OSPF

Purpose

Use the IPv6_OSPF statement to specify various parameters that apply globally to IPv6 OSPF, either to all interfaces or to the overall IPv6 OSPF autonomous system.



Operands

RouterID

Every router in an IPv6 OSPF routing domain must be assigned a unique 32-bit router ID.

The value used for the IPv6 OSPF router ID is chosen as follows:

- If this parameter is configured, the value configured is used as the IPv6 OSPF router ID.
- If this parameter is not configured and IPv4 OSPF is also active on MPServer, then the IPv4 Router ID value will also be used for IPv6.

Valid values are any 32-bit value, in dotted decimal format (in other words, specified as an IPv4-style IP address).

Restriction: If IPv4 OSPF is NOT active, then RouterID is a required configuration parameter.

Comparison

Tells MPRoute where external routes fit in the IPv6 OSPF hierarchy. IPv6 OSPF supports two types of external metrics. Type1 external metrics are equivalent to the link state metric. Type2 external metrics are greater than the cost of any path internal to the AS. Use of Type2 external metrics assumes that routing between autonomous systems is the major cost of routing a packet, and eliminates the need for conversion of external costs to internal link state metrics. Valid values are Type1 (or 1) or Type2 (or 2).

For more information about the COMPARISON configuration parameter, see [“Notes on defining route precedence in a multiprotocol environment”](#) on page 216.

Demand_Circuit

Global demand circuit setting. Coding YES enables demand circuits for IPv6 OSPF. Demand circuit parameters can then be coded on the IPv6_OSPF_Interface statement. Valid values are Yes or No.

Instance

Provides the default instance number for MPRoute. MPRoute supports only one instance of IPv6 OSPF on a link, and this parameter specifies the default value for all IPv6 OSPF interfaces. This value can be overridden on individual IPv6_OSPF_Interface statements. Valid values are any integer from 0 to 255.

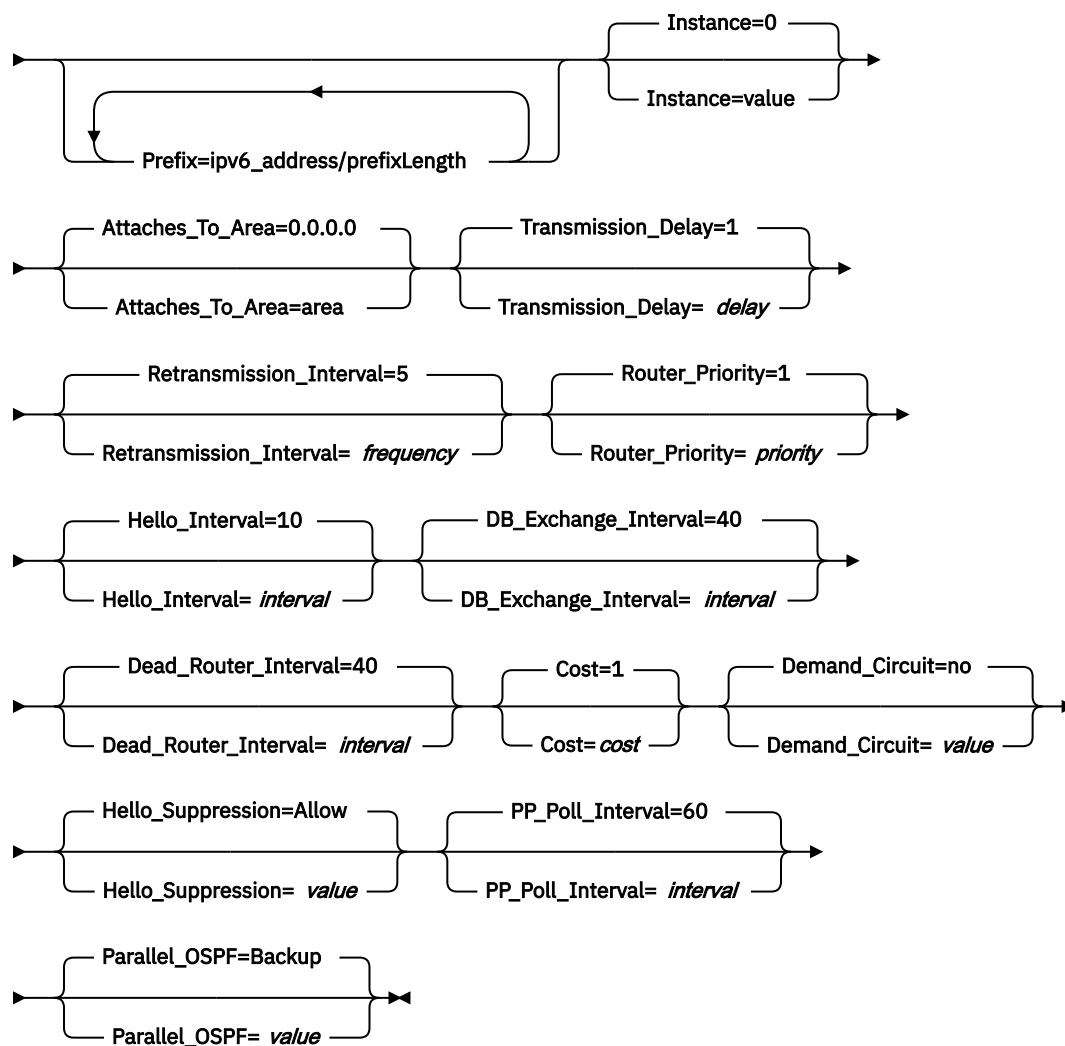
DR_Max_Adj_Attempt

Specifies the maximum number of adjacency attempts to be used for reporting and controlling futile neighbor state loops. After the adjacency attempt count for a neighboring designated router reaches the threshold, an informational message is issued to report the problem. If a redundant interface that can reach the neighbor is available, adjacency formation is attempted over that interface. An informational message is issued to report the interface switch and adjacency formation attempt. Valid values are in the range 0–100. The value 0 specifies infinite retries. For more information, see [“Preventing futile neighbor state loops during adjacency formation”](#) on page 198.

IPv6_OSPF_INTERFACE**Purpose**

Sets the IPv6 OSPF parameters for the TCP/IP network interfaces. This statement needs to be replicated in the configuration file for each IPv6 interface over which OSPF will operate.

➤ IPv6_OSPF_Interface — Name=*interface_name* ➔



Operands

Name

The name of the interface.

This name must match the interface name coded on the LINK statement in the TCP/IP configuration file. Valid values are any character string of 1-16 characters in length, except numeric interface names containing a decimal point (for example, 123.456). Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv6 interfaces”](#) on page 212.

Prefix

Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, MPRoute will add a direct route to the prefix identified by the first *prefixLength* bits of *ipv6_address*. Valid values for *ipv6_address* are any colon-hexadecimal IPv6 address. Valid values for *prefixLength* are any integer value from 1 to 127. The prefix identified by the first *prefixLength* bits of *ipv6_address* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If routers on the link are advertising prefixes using either IPv6 OSPF or IPv6 Router Discovery, prefixes being advertised as on-link by the routers should not be configured using this keyword. However, if IPv6 Router Discovery or IPv6 OSPF is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the prefix is configured using this keyword and is also advertised by a router as being on-link, the route in the TCP/IP stack's route table will be the route added by MPRoute as a result of this keyword being specified. Any route for the same prefix that is learned from IPv6 OSPF or Router Discovery is ignored as long as the MPRoute-configured route exists.

Instance

Specifies the IPv6 protocol instance number for this interface. This value should be the same as the instance value of other IPv6 OSPF hosts or routers that MPRoute will be communicating with on this link. This value will be set on all outgoing IPv6 OSPF packets, and all incoming IPv6 OSPF packets whose instance value does not match the value coded for this interface will be ignored. This permits multiple instances of OSPF to be run on this link. MPRoute only supports one instance per link, however by coding this parameter MPRoute can interact with other routers that may support multiple instances. This value will default to the value coded on the Instance parameter of the IPv6_OSPF configuration statement. If that value is not coded, the default is 0. Valid values are 0-255.

Attaches_To_Area

IPv6 OSPF area to which this interface attaches. Valid values are 0.0.0.0 (the backbone), or any area defined by the IPv6_AREA statement.

Retransmission_Interval

Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from 1 to 65535 seconds.

If this parameter is set too low, needless retransmissions will occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

Transmission_Delay

This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the interface. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent out from this interface, it will be aged by this configured transmission delay. Valid values are 1 to 65535 seconds.

Router_Priority

This value is used for multiaccess networks to elect the designated router, with the highest priority router being elected. Valid values are 0 to 255. A value of 0 indicates that MPRoute cannot become designated router.

A value of 1 indicates the lowest possible eligible priority and a value of 255 indicates the highest possible priority. A value of 0 indicates that MPRoute is not eligible to be a designated router on this link.

Hello_Interval

This parameter defines the number of seconds between IPv6 OSPF Hello packets being sent out this interface. This value must be the same for all routers attached to a common link. Valid values are 1 to 255 seconds.

DB_Exchange_Interval

The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure will be restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval will be set to the Dead_Router_Interval. Valid values are 2 through 65535.

Dead_Router_Interval

The interval in seconds, after not having received an IPv6 OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Setting this value too close to the Hello_Interval can result in the collapse of adjacencies. A value of 4*Hello_Interval is recommended. This value must be the same for all routers attached to a common link. Valid values are 2 to 65535.

Cost

The OSPF cost for this interface. The cost is used to determine the shortest path to a destination. Valid values are 1 to 65535.

Demand_Circuit

This parameter, when coded with YES, causes Link State Advertisements (LSAs) to not be periodically refreshed over this interface. Only LSAs with real changes will be advertised. In addition, coding this parameter to YES causes LSAs flooded over this interface to never age out. Valid values are YES or NO. For more information on the Demand_Circuit=YES and related topics, such as handling high cost links, see [“Notes on managing high-cost links”](#) on page 215.

Hello_Suppression

This parameter is meaningful only for demand circuits. This parameter allows you to configure the interface to request Hello_Suppression. This parameter is used only on point-to-point. Valid values are ALLOW, REQUEST, or DISABLE.

If either or both sides specify DISABLE, Hello_Suppression is disabled. If both specify ALLOW, Hello_Suppression is disabled. If one specifies ALLOW and the other REQUEST, or if both specify REQUEST, Hello_Suppression is enabled.

PP_Poll_Interval

This parameter specifies the interval (in seconds) that MPRoute should use when attempting to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available. This parameter is meaningful only if Demand_Circuit is coded YES and Hello_Suppression has been enabled. Valid values are 0 to 65535.

Parallel_OSPF

This parameter designates whether the IPv6 OSPF interface is primary or backup when more than one IPv6 OSPF interface is defined to the same link. Only one of these interfaces can be configured as primary, meaning that it will be the interface to carry the IPv6 OSPF protocol traffic between MPRoute and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If the primary interface is later reactivated, IPv6 OSPF traffic will not be automatically switched back from the backup interface to the primary interface. If you want to switch OSPF traffic back to the primary interface, the backup interface must be stopped. If none of the interfaces to the common subnet are configured as primary, a primary interface will be selected by MPRoute. Valid values are Backup and Primary.

Tip: For IPv6, MPRoute considers two interfaces to be on the same link if they have any prefixes in common.

IPv6_RANGE**Purpose**

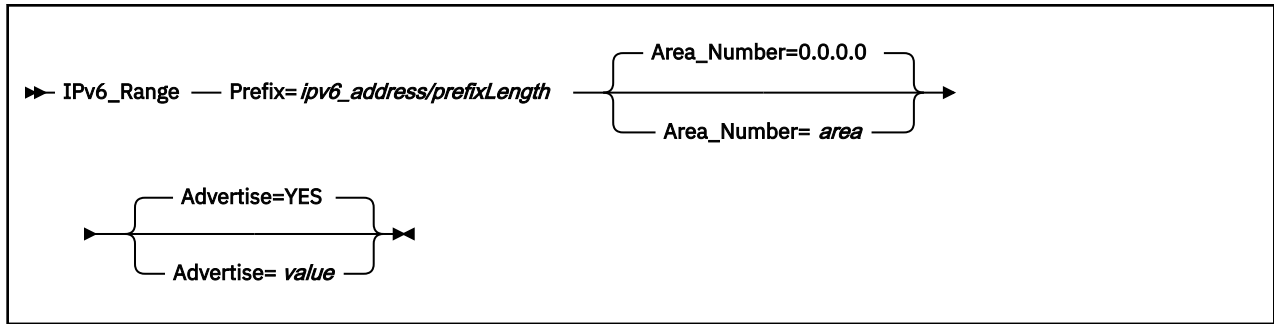
Adds ranges to IPv6 OSPF areas. External to the area, a single route is advertised for each address range. For example, if an IPv6 OSPF area were to consist of the prefix 2001:0db8:1:2::/64, all addresses falling within that prefix would be defined as consisting of a single address range. The address range would be specified as an address of 2001:0db8:1:2:: together with a prefix length of 64. Outside of the area, all addresses that fall within that prefix would be advertised as a single route to prefix 2001:0db8:1:2::/64.

Ranges can be defined to control which routes are advertised external to an area.

There are two choices:

- When IPv6 OSPF is configured to advertise the range, a single interarea route is advertised for the range if at least one component route of the range is active within the area.
- When IPv6 OSPF is configured not to advertise the range, no interarea prefix routes are advertised for routes that fall within the range. Ranges cannot be used for areas that serve as transit areas for virtual links. Also, when ranges are defined for an area, IPv6 OSPF will not function correctly if the area is partitioned but is connected by the backbone.

Ranges cannot be used for areas that serve as transit areas for virtual links. Also, when ranges are defined for an area, OSPF will not function correctly if the area is partitioned but is connected by the backbone.



Operands

Prefix

Common prefix of IP addresses in this range, with the prefix length.

Area_Number

Area number for which to add this range. Valid values are any defined areas.

Advertise

Specifies whether this range will be advertised to other areas. Valid values are YES or NO.

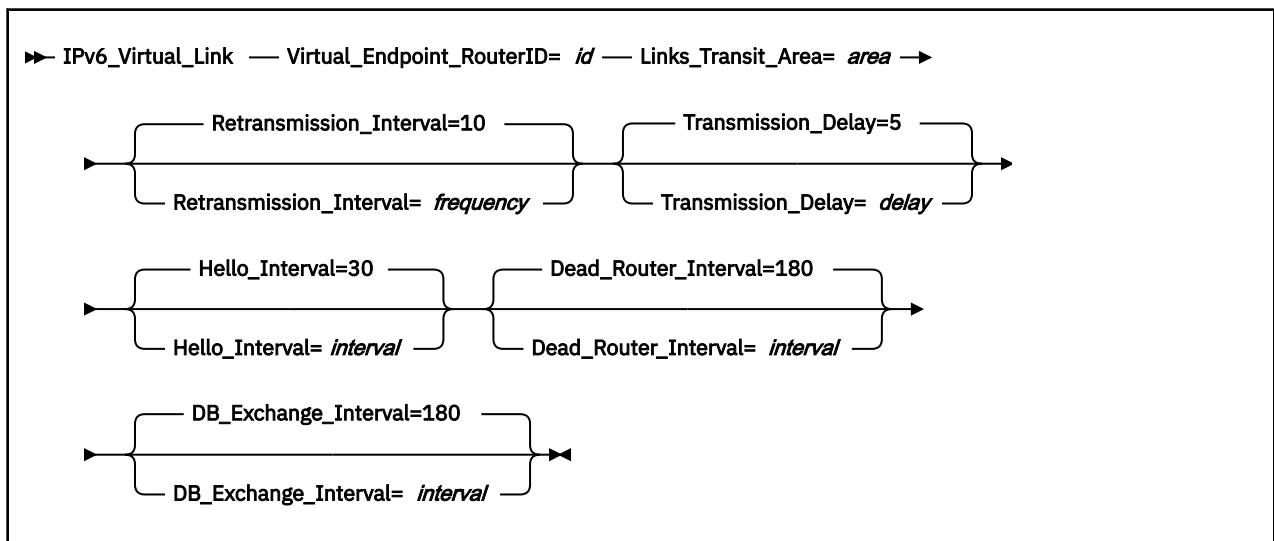
IPv6_VIRTUAL_LINK

Purpose

Configures a virtual link between two area border routers. To maintain backbone connectivity you must have all of your backbone routers interconnected either by permanent or virtual links. Virtual links are considered to be separate router interfaces connecting to the backbone area. Therefore, you are asked to specify many of the interface parameters when configuring a virtual link.

Virtual links can be configured between any two backbone routers that have an interface to a common nonbackbone, nonstub area. Virtual links are used to maintain backbone connectivity and must be configured at both endpoints.

Tip: OSPF virtual links are not to be confused with Virtual IP Address support (VIPA).



Operands

Virtual_Endpoint_RouterID

32-bit IPv6 OSPF router ID of the virtual neighbor (other endpoint), specified in dotted-decimal notation.

Links_Transit_Area

This is the nonbackbone, nonstub area through which the virtual link is configured. Virtual links can be configured between any two area border routers that have an interface to a common nonbackbone and nonstub area. Virtual links must be configured in each of the link's two endpoints. Valid values are 0.0.0.1 to 255.255.255.255.

Retransmission_Interval

Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from 1 to 65535 seconds.

If this parameter is set too low, needless retransmissions will occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

Transmission_Delay

This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the virtual link. Each link-state advertisement has a finite lifetime of one hour. As each link-state advertisement is sent out from this virtual link, it will be aged by this configured transmission delay. Valid values are 1 to 65535 seconds.

Hello_Interval

This parameter defines the number of seconds between OSPF Hello packets being sent out from this virtual link. Valid values are 1 to 255 seconds. The Hello_Interval should be set higher than the same value used on the intervening, actual IPv6 OSPF interfaces.

Dead_Router_Interval

The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Valid values are 2 to 65535. The dead router interval should be set higher than the same value used on the intervening, actual, IPv6 OSPF interfaces.

DB_Exchange_Interval

The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure will be restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval will be set to the Dead_Router_Interval. Valid values are 2 through 65535.

IPv6 RIP configuration statements

This section contains descriptions of the following IPv6 RIP configuration statements:

- IPV6_ACCEPT_RIP_ROUTE
- IPV6_RIP_FILTER
- IPV6_IGNORE_RIP_NEIGHBOR
- IPV6_ORIGINATE_RIP_DEFAULT
- IPV6_RIP_INTERFACE
- IPV6_RIP_SEND_ONLY

IPv6_ACCEPT_RIP_ROUTE

Purpose

Allows a prefix or host route to be accepted independent of whether the interface it was received on has the corresponding reception parameter enabled (prefix or host). Routes added in this manner can be thought of as a list of exception conditions.

Note: Coding this statement will not enable updates for this destination to be received on IPv6 RIP interfaces with RECEIVE_RIP=NO coded.

```
➤ IPv6_Accept_RIP_Route — IP_address= address ➤
```

Operands

IP_address

Destination route to be unconditionally accepted, specified in colon-hexadecimal format.

IPv6_RIP_FILTER

Purpose

Allows for the specification of routes that are not to be sent or received over IPv6 RIP interfaces. The IPv6_RIP_Filter statement can be coded stand-alone in the MPRoute configuration file (nosend and noreceive only) to apply to all configured IPv6 RIP interfaces.

```
➤ IPv6_RIP_Filter= (type,dest/prefix_len) ➤
```

Operands

type

The type can be any of the following values:

nosend

Specifies that routes matching the dest and prefix_len are not to be sent over IPv6 RIP interfaces. This option serves as an IPv6 RIP output filter.

noreceive

Specifies that routes matching the dest and prefix_len are to be ignored in messages received over IPv6 RIP interfaces. This option serves as an IPv6 RIP input filter.

dest

The dest specifies the destination route in colon-hexadecimal format. Alternatively, an asterisk (*), which matches *any* IPv6 destination, can be coded to filter out all routes sent or received over an interface. The use of the asterisk is also referred to as a blackhole filter. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: The blackhole nosend filter will not filter out the sending of the default route when the IPv6_Originate_RIP_Default statement is also configured.

prefix_len

The prefix_len specifies the number of significant bits in the destination to be filtered. If not coded, the default prefix_len will be 128, meaning apply the filter to the dest as coded. Coding the prefix_len has no meaning and is not valid if the dest is coded as an asterisk (*) for a blackhole filter.

IPv6_IGNORE_RIP_NEIGHBOR

Purpose

Specifies that IPv6 RIP routing table messages from this gateway are to be ignored. This option serves as an IPv6 RIP input filter.

➡ IPv6_Ignore_Rip_Neighbor — IP_address= *link_local_address* ➡

Operands

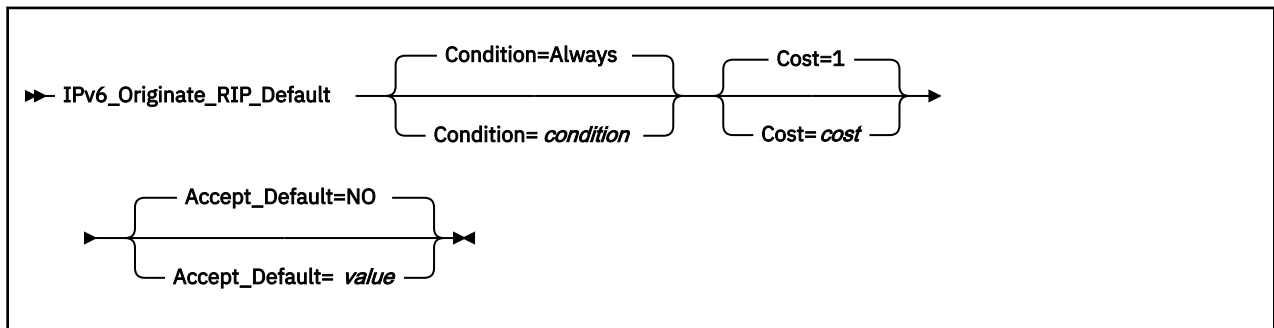
IP_address

Specifies the link-local IP address, in colon-hexadecimal format, of the gateway from which routing table messages will be ignored. For multiple IP addresses, the statement must be repeated for each IP address.

IPv6_ORIGINATE_RIP_DEFAULT

Purpose

Indicates under what conditions IPv6 RIP will support Default route (destination/prefix_len ::/0) generation.



Operands

Condition

Condition for when IPv6 RIP is to advertise this router as a default router. Valid values are:

Always

Always originate IPv6 RIP default.

Never

Never advertise this router as a default IPv6 RIP router.

OSPF

Advertise this router as a default IPv6 RIP router if there are any IPv6 OSPF routes available.

Cost

Specifies the cost that IPv6 RIP will advertise with the default route that it originates. Valid values are 1 to 16.

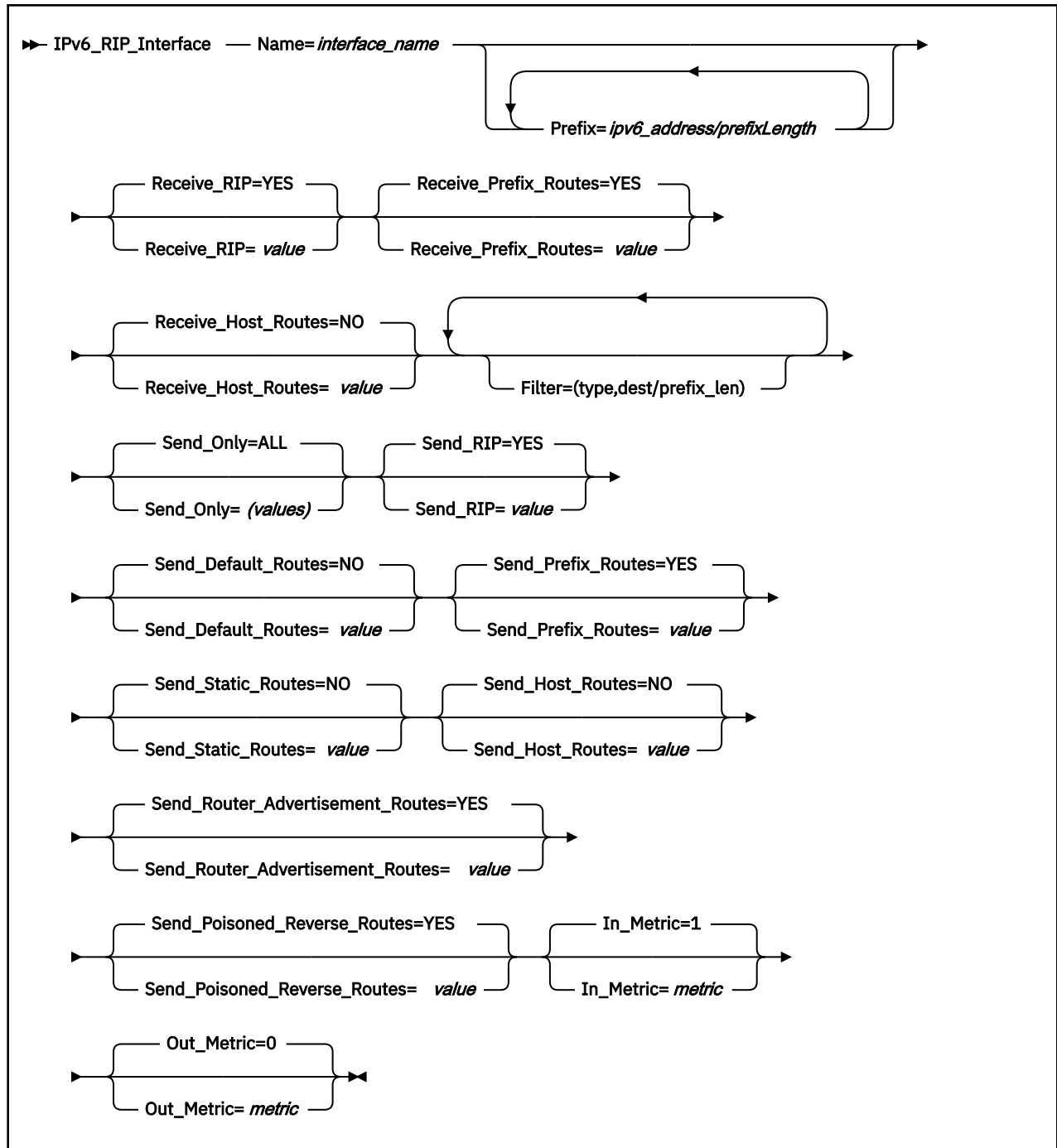
Accept_Default

If IPv6 RIP learns of a default route that has a lower cost than is specified in this statement, that default route will be accepted and replace this router's default route. Additionally, if this parameter is coded to YES, then IPv6 RIP will learn default routes from inbound IPv6 RIP packets, even if their cost is higher than default routes originated by this host. Valid values are YES or NO.

IPv6_RIP_INTERFACE

Purpose

Configures the IPv6 RIP parameters for each IP interface. This statement will need to be replicated in the configuration file for each IP interface over which IPv6 RIP will operate.



Operands

Name

The name of the interface.

This name must match the interface name coded on the LINK statement in the TCP/IP configuration file. Valid values are any character string of 1-16 characters in length, except numeric interface names containing a decimal point (for example, 123.456). Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv6 interfaces”](#) on page 212.

Prefix

Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, MPRoute will add a direct route to the prefix identified by the first *prefixLength* bits of *ipv6_address*. Valid values for *ipv6_address* are any valid colon-hexadecimal IPv6 address. Valid values for *prefixLength* are any integer value from 1 to 127. The prefix identified by the first *prefixLength* bits of *ipv6_address* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If IPv6 Router Discovery is in use by the routers on the link, prefixes being advertised as on-link by the routers by way of Router Discovery should not be configured using this keyword. However, if IPv6 Router Discovery is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the same prefix is configured using this keyword and learned from Router Discovery, the route in the TCP/IP stack's route table will be the route added by MPRoute as a result of this keyword being specified. Any route for the same prefix that is learned from Router Discovery will be ignored as long as the MPRoute route exists.

Receive_RIP

Specifies whether IPv6 RIP updates will be accepted over this interface. Valid values are:

YES

IPv6 RIP packets will be received over this interface, subject to other filters. This is the default value.

NO

No IPv6 RIP packets will be received over this interface, regardless of any other filters.

Receive_Prefix_Routes

Specifies whether or not to learn routes for prefixes over this interface. If this is not set, only prefixes explicitly allowed using the IPv6_Accept_RIP_Route configuration statement will be accepted on this interface. Valid values are YES or NO.

Receive_Host_Routes

Specifies whether or not to learn routes for hosts over this interface. If this is not set, only hosts explicitly allowed using the IPv6_Accept_RIP_Route configuration statement will be accepted on this interface. Valid values are YES or NO.

filter

Multiple filter parameters can be coded on a IPv6_RIP_Interface statement. When specified on the IPv6_RIP_Interface statement, the filter parameter applies only to the corresponding IPv6 RIP interface. The IPv6_RIP_Filter statement can also be coded stand-alone in the MPRoute configuration file (nosend and noreceive only) to apply to all configured IPv6 RIP interfaces.

The type can be any of the following values:

Value

Description

nosend

Specifies that routes matching the dest and prefix_len are not to be sent over this interface. This option serves as an IPv6 RIP output filter.

noreceive

Specifies that routes matching the dest and prefix_len are to be ignored in messages received over this interface. This option serves as an IPv6 RIP input filter.

send

Specifies that routes matching the dest and prefix_len are to be sent over only this interface (or any other IPv6 RIP interface with an equivalent filter). This option serves as an IPv6 RIP output filter and can be used for inbound and outbound traffic splitting.

send_cond

Specifies that routes matching the dest and prefix_len are to be sent over only this interface when this interface is active (or any other active IPv6 RIP interface with an equivalent filter). If this interface is inactive, the routes can be sent over other interfaces. This option serves as an IPv6 RIP output filter and can be used for inbound and outbound traffic splitting.

receive

Specifies that routes matching the dest and prefix_len are to be received over only this interface (or any other IPv6 RIP interface with an equivalent filter). If received over other IPv6 RIP interfaces, the routes are discarded. This option serves as an IPv6 RIP input filter.

receive_cond

Specifies that routes matching the dest and prefix_len are to be received over only this interface when this interface is active (or any other active IPv6 RIP interface with an equivalent filter). If this interface is inactive, the routes can be received over all other active IPv6 RIP interfaces. This option serves as an IPv6 RIP input filter.

The dest specifies the destination route in colon-hexadecimal format. Alternatively, an asterisk (*) can be coded in conjunction with the nosend and noreceive filter types. This serves as a blackhole filter that can be used to filter out all routes sent or received over an interface. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: If the blackhole nosend filter is used, it will not filter out the sending of the default route when the Originate_RIP_Default statement is also configured.

The prefix_len specifies the number of significant bits in the destination to be filtered. If not coded, the default prefix_len will be 128, meaning apply the filter to the dest route as coded. Coding the prefix_len has no meaning and is not valid if the dest is coded as an asterisk (*) for a blackhole filter.

Send_Only

Specifies send restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL

Specifies no send restrictions.

DEFAULT

Sends the default route.

DIRECT

Sends direct routes.

TRIGGERED

Only sends routes when requested or when a route becomes inactive (metric 16).

DEFAULT, DIRECT, and TRIGGERED are OR'd together to determine what should be sent. Thus, coding SEND_ONLY=(DEFAULT,DIRECT) will send the default route and direct routes. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the IPv6_RIP_Interface statement, the Send_Only parameter applies only to the corresponding IPv6 RIP interface. The IPv6_RIP_Send_Only statement can also be coded stand-alone in the MPRoute configuration file to apply to all IPv6 RIP interfaces.

Send_RIP

Specifies whether or not IPv6 RIP advertisements will be sent over this interface. Valid values are YES or NO.

Send_Default_Routes

Advertise the default route (destination/prefix_len ::/0), if it is available, in IPv6 RIP responses sent from this IP source address. Valid values are YES or NO. If DEFAULT is coded on the Send_Only parameter or the stand-alone IPv6_RIP_Send_Only statement, the Send_Default_Routes parameter is ignored and will be set to YES.

Send_Prefix_Routes

Advertise all prefix routes in IPv6 RIP responses sent from this IP address. Valid values are YES or NO.

Send_Static_Routes

Advertise static and direct routes in IPv6 RIP responses sent from this IP source address. Split horizon is applied; that is, static routes configured over an interface will not be included in IPv6 RIP responses sent from that interface. Valid values are YES or NO.

Send_Host_Routes

Advertise host routes in IPv6 RIP responses sent from this IP source address. In this context, a host route is one with a prefix length of 128. Valid values are YES or NO.

Send_Router_Advertisement_Routes

Advertise router advertisement routes in IPv6 RIP responses sent from this IP source address. These are routes that have been learned by the stack using IPv6 Router Discovery and that MPRoute has learned from the stack. Split horizon is applied; that is, router advertisement routes learned over an interface will not be included in IPv6 RIP responses sent from that interface. Valid values are YES or NO.

Send_Poisoned_Reverse_Routes

Advertise poisoned reverse routes over the interface corresponding to the next hop. A poison reverse route is one with an infinite metric (16). Valid values are YES or NO. If NO is specified, MPRoute still uses split horizon.

In_Metric

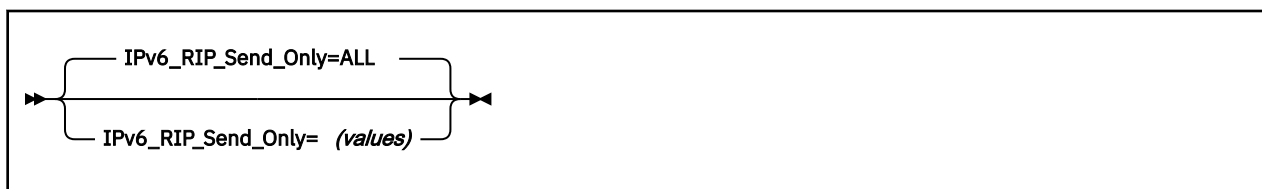
Specifies the value of the metric to be added to IPv6 RIP routes received over this interface prior to installation in the routing table. Valid values are 1 to 15.

Out_Metric

Specifies the value of the metric to be added to IPv6 RIP routes advertised over this interface. Valid values are 0 to 15.

IPv6_RIP_SEND_ONLY**Purpose**

Allows for the specification of the types of routes that are to be included in advertisements sent over IPv6 RIP interfaces. The IPv6_RIP_Send_Only statement can be coded stand-alone in the MPRoute configuration file to apply to all IPv6 RIP interfaces.

**Operands****(values)**

Specifies send restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL

Specifies no send restrictions.

DEFAULT

Sends the default route.

DIRECT

Sends direct routes.

TRIGGERED

Only sends routes when requested or when a route becomes inactive (metric 16).

DEFAULT, DIRECT, and TRIGGERED are OR'd together to determine what should be sent. Thus, coding SEND_ONLY=(DEFAULT,DIRECT) will send the default route and direct routes. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the IPv6_RIP_Send_Only statement in the MPRoute configuration file, this statement applies to all IPv6 RIP interfaces. The SEND_ONLY parameter can also be coded on the IPv6_RIP_Interface statement. When specified on the IPv6_RIP_Interface statement, the SEND_ONLY parameter applies only to the corresponding IPv6 RIP interface.

Common configuration statements for RIP and OSPF

This section contains descriptions of the common configuration statements:

- DEFAULT_ROUTE
- INTERFACE
- GLOBAL_OPTIONS
- IPV6_DEFAULT_ROUTE
- IPV6_INTERFACE

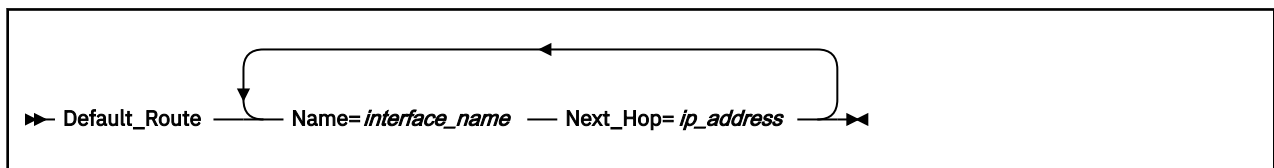
DEFAULT_ROUTE

Purpose

Allows IPv4 default routes to be specified to MPRoute. Default routes are created using any of the following:

- GATEWAY statement
- Default_Route statement
- Learned by routing protocol

Up to 16 default routes can be configured using this Default_Route statement. The Send_Default_Routes keyword on the RIP_Interface statement indicates whether or not to advertise the default routes over that interface.



Operands

Name

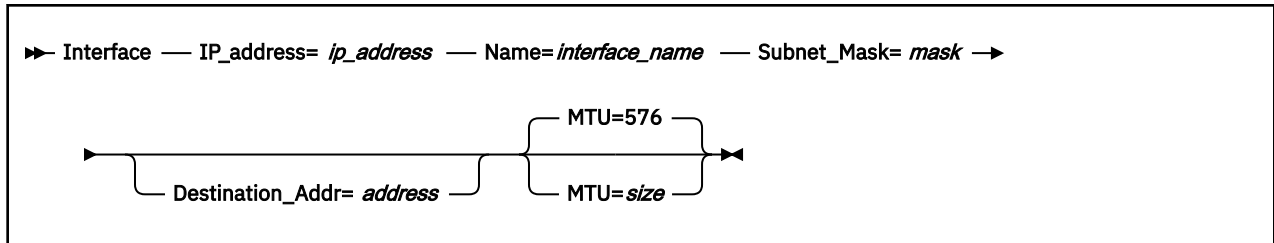
The name of the interface used in the default route. This name must match a link name coded on the HOME statement in the TCP/IP configuration file. Valid values are any 16 characters, except numeric interface names containing a decimal point (for example, 123.456). VIPA interfaces cannot be used as the *interface_name*.

Next_Hop

IP address of the next hop used in the default route.

INTERFACE**Purpose**

Allows certain values to be specified for generic IPv4 interfaces, which are interfaces that are neither OSPF nor RIP interfaces. Each IPv4 interface that is neither an OSPF nor an RIP interface should be configured to MPRoute using the INTERFACE statement unless it is a non-point-to-point interface and the default values for Subnet_Mask and MTU are acceptable for that interface.

**Operands****IP_address**

The IP address can be a valid IP address that is configured on the system or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are below. The result of coding a wildcard value is that all configured interfaces whose IP address matches the wildcard will be configured as interfaces. Configured interface IP addresses and names will be matched against possible wildcards in the order they appear below with the name and any matching wildcard being the best match, x.y.z.* being second best, and so forth.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Name

The name of the interface. This name must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP configuration file. Note that numeric interface names containing a decimal point (for example, 123.456) are not allowed. If an exact IP address match is not found, then this parameter is used first when searching wildcard addresses. If a definition with matching name and any matching wildcard is not found, then the most specific wildcard address that matches is used. The same wildcard address can be configured more than once with unique names.

Subnet_Mask

Subnet mask for the associated interface's IP address. For more information, see [“Notes on defining IPv4 interfaces”](#) on page 207.

Destination_Addr

IP address of the host at the remote end of this interface. This parameter is only valid for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface will not be added to the TCP/IP route table. A subnet route for the interface will be added at MPRoute initialization independent of whether this parameter is specified.

MTU=size

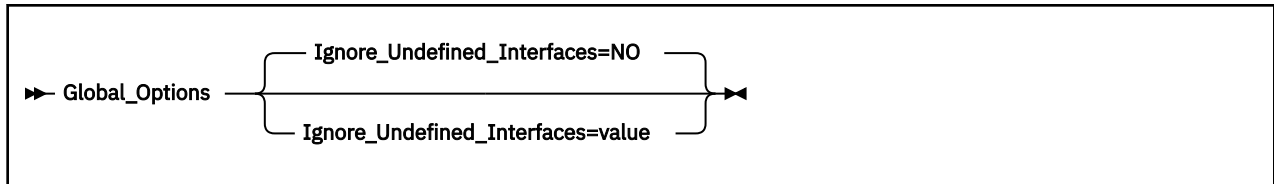
Specifies the MTU size for MPRROUTE to add to the routing table for routes that take this interface. The size must be 0, or a number in the range of 576 to 65535.

If the LINK statement in TCP/IP configuration file associated with *interface_name* has an MTU size (MTU *mtusize*), specify 0; a non-zero value is ignored.

GLOBAL_OPTIONS

Purpose

Use the GLOBAL_OPTIONS statement to configure miscellaneous options to MPRoute which apply to OSPF, RIP, or neither.



Operands

Ignore_Undefined_Interfaces

Instructs MPRoute on how to handle stack interfaces that are not configured by way of OSPF_INTERFACE, RIP_INTERFACE, IPV6_RIP_INTERFACE, IPV6_OSPF_INTERFACE, INTERFACE, or IPV6_INTERFACE statements, either explicit or wildcard. NO indicates that MPRoute will configure such interfaces with default values (including using the class mask as the subnet mask and overriding stack definition values with these default values for IPv4 interfaces), and possibly advertise these interfaces to the rest of the network if OSPF or RIP filters permit it. YES indicates that MPRoute will ignore these interfaces, will not configure them, and will not advertise them under any circumstances. IF YES is coded, MPRoute will not advertise these interfaces or their attached subnets or prefixes, and it will not update any stack definition values. Static routes coded in the TCP/IP configuration file over interfaces that are ignored by MPRoute will still be advertised by MPRoute into OSPF and/or RIP, if appropriate filters and settings permit advertisement of static routes.

IPv6_DEFAULT_ROUTE

Purpose

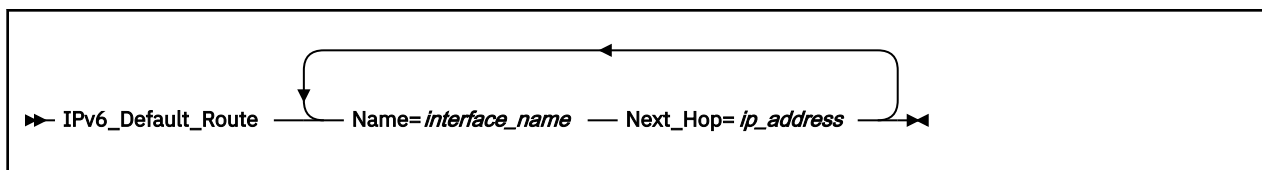
Allows IPv6 default routes to be specified to MPRoute. IPv6 default routes are created using any of the following:

- GATEWAY Statement
- IPv6_Default_Route statement
- Learned by routing protocol
- Router advertisements

When IPv6 default routes are specified using more than one of these methods, the method used to create the default routes is determined according to the following list, in order of descending precedence:

1. Default routes specified using the GATEWAY statement
2. Default routes learned by routing protocol
3. Default routes specified using the IPV6_DEFAULT_ROUTE statement
4. Router advertisements

Up to 16 default routes can be configured using this IPV6_DEFAULT_ROUTE statement. The SEND_DEFAULT_ROUTES keyword on the IPV6_DEFAULT_ROUTE statement indicates whether or not to advertise the IPv6 default routes over that interface.



Operands

Name

The name of the interface used in the default route. This name must match an interface name coded on the LINK statement in the TCP/IP configuration file. Valid values are any 16 characters, except numeric interface names containing a decimal point (for example, 123.456).

Next_Hop

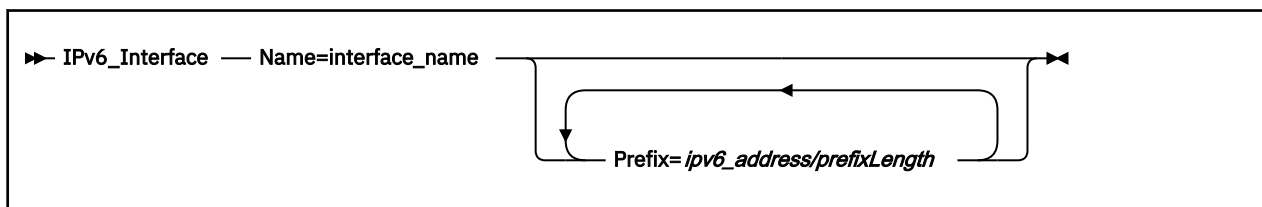
IP address of the next hop used in the default route, in colon-hexadecimal format. This IP address must be reachable using a direct route over the specified interface. If it is not, this next hop will not be installed.

IPv6_INTERFACE

Purpose

Allows certain values to be specified for generic IPv6 interfaces, which are interfaces that are neither IPv6 OSPF nor IPv6 RIP interfaces. If GLOBAL_OPTIONS is coded with IGNORE_UNDEFINED_INTERFACES=YES, then IPv6 interfaces that will not be used for routing but which MPRoute should be aware of should be coded in the MPRoute configuration file. If that option is not coded, it is not necessary to code all non-routing IPv6 interfaces to MPRoute if default values are acceptable and you do not need to code additional prefixes on the interface.

Tip: Use the SMSG *server_id* GENERIC6 command to display information about IPv6_INTERFACES.



Operands

Name

The name of the interface. This name must match the interface name coded on the LINK statement in the TCP/IP configuration file. Valid values are any character string of 1-16 characters in length, except numeric interface names containing a decimal point (for example, 123.456). Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.

Tip: For more information about how wildcard interfaces are parsed, see [“Notes on defining IPv6 interfaces”](#) on page 212.

Prefix

Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, MPRoute will add a direct route to the prefix identified by the first *prefixLength* bits of *ipv6_address*. Valid values for *ipv6_address* are any valid colon-hexadecimal IPv6 address. Valid values for *prefixLength* are any integer value from 1 to 127. The prefix identified by the first *prefixLength* bits of *ipv6_address* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If IPv6 Router Discovery is in use by the routers on the link, prefixes being advertised as on-link by the routers by way of Router Discovery should not be configured using this keyword.

However, if IPv6 Router Discovery is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the same prefix is configured using this keyword and learned from Router Discovery, the route in the TCP/IP stack's route table will be the route added by MPRoute as a result of this keyword being specified. Any route for the same prefix that is learned from Router Discovery will be ignored as long as the MPRoute route exists.

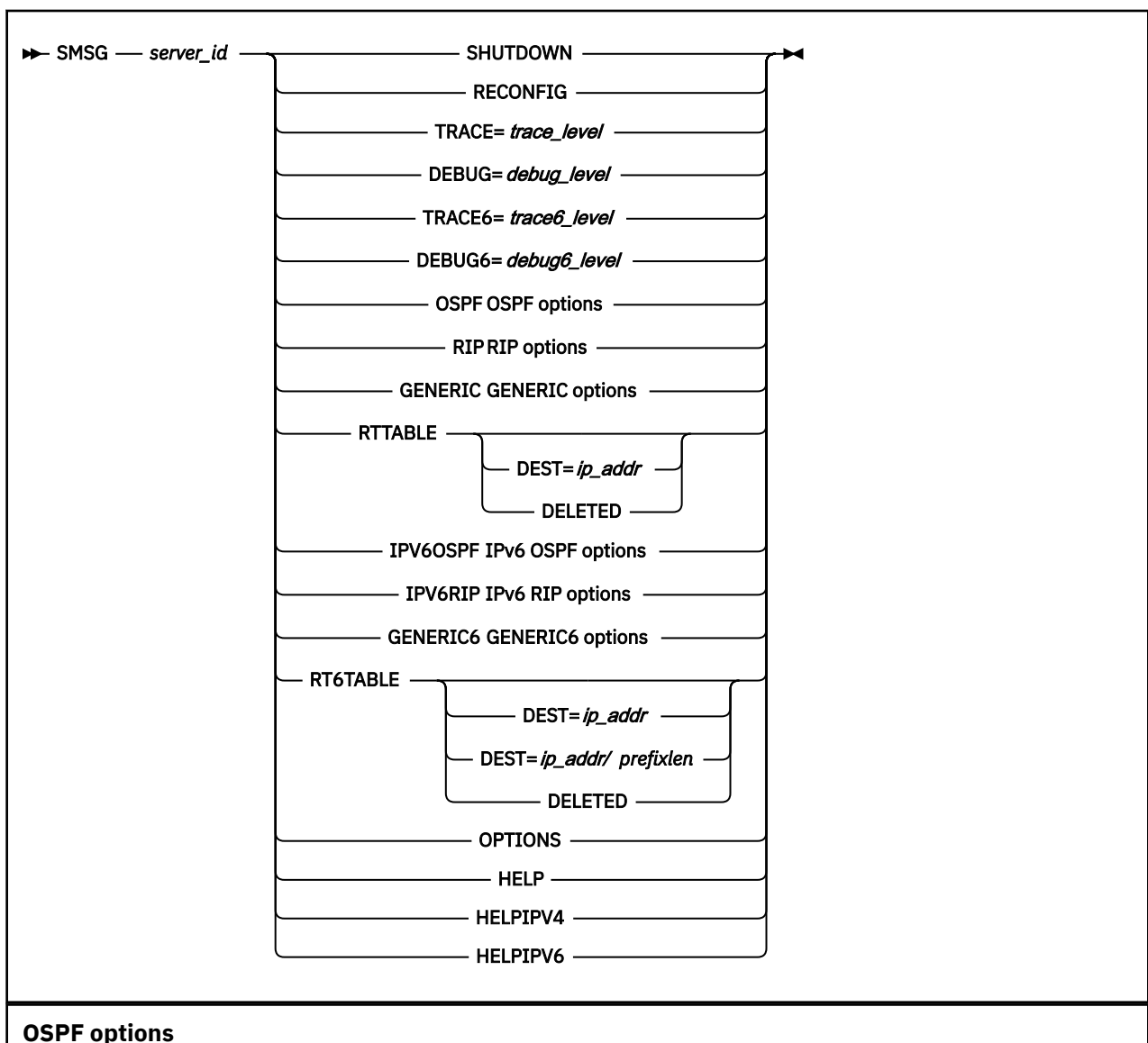
Dynamic Server Operation

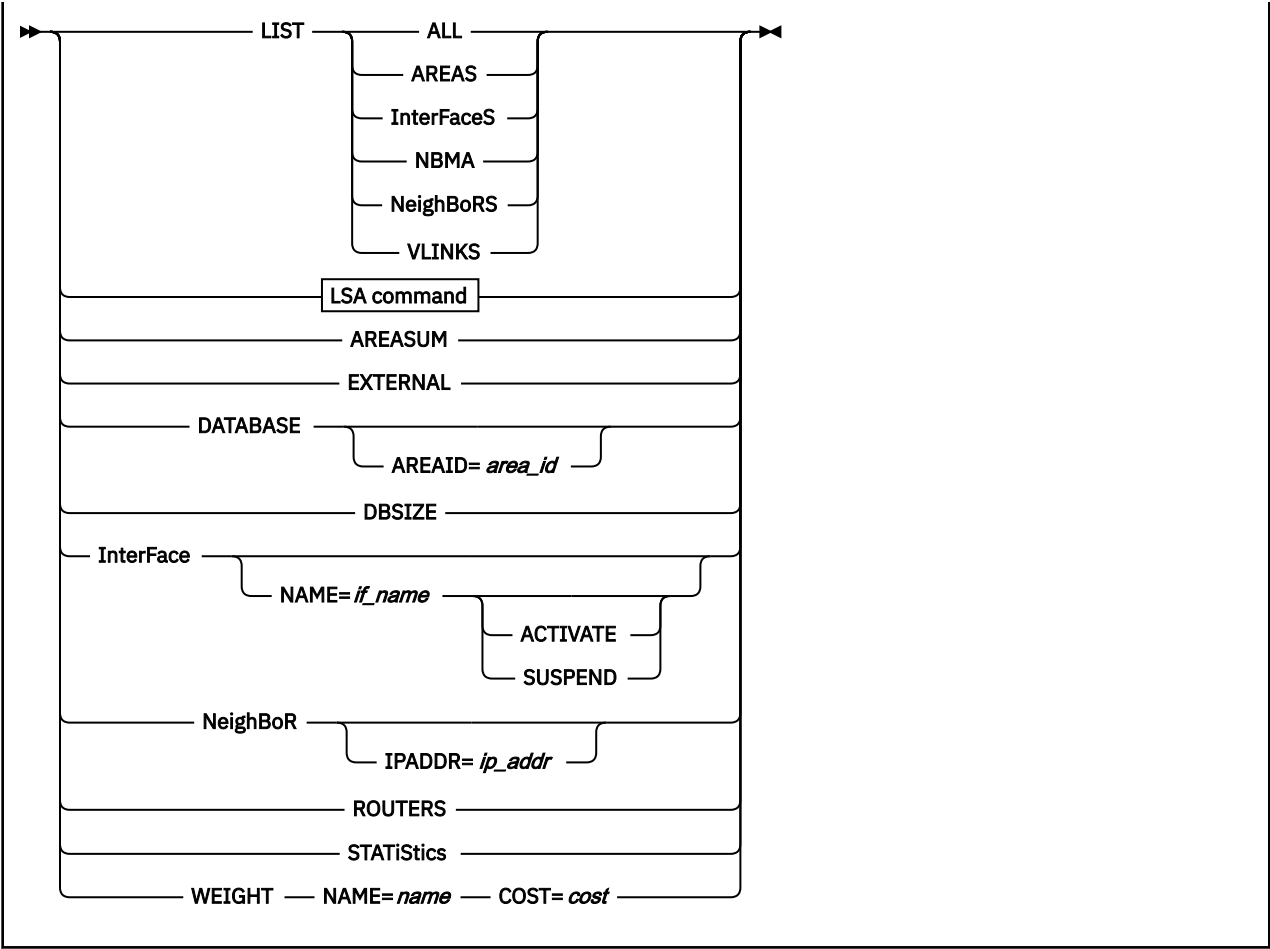
The VM Special Message Facility (MSG) command provides an interactive interface to the MPRoute virtual machine to perform privileged system administration tasks.

Privileged users are specified in the OBEY list of the TCP/IP server configuration file.

Note: Command responses are returned to the originator of the command through CP MSG commands.

SMSG Interface to the MPRoute Server





LSA command

➤ LSA — LSTYPE= *ls_type* — LSID= *lsid* — ORIGINator= *ad_router* — AREAID= *area_id* —

RIP options

➤ LIST — ALL — InterFaceS — ACCEPTED — InterFace — NAME= *if_name* — FILTERS —

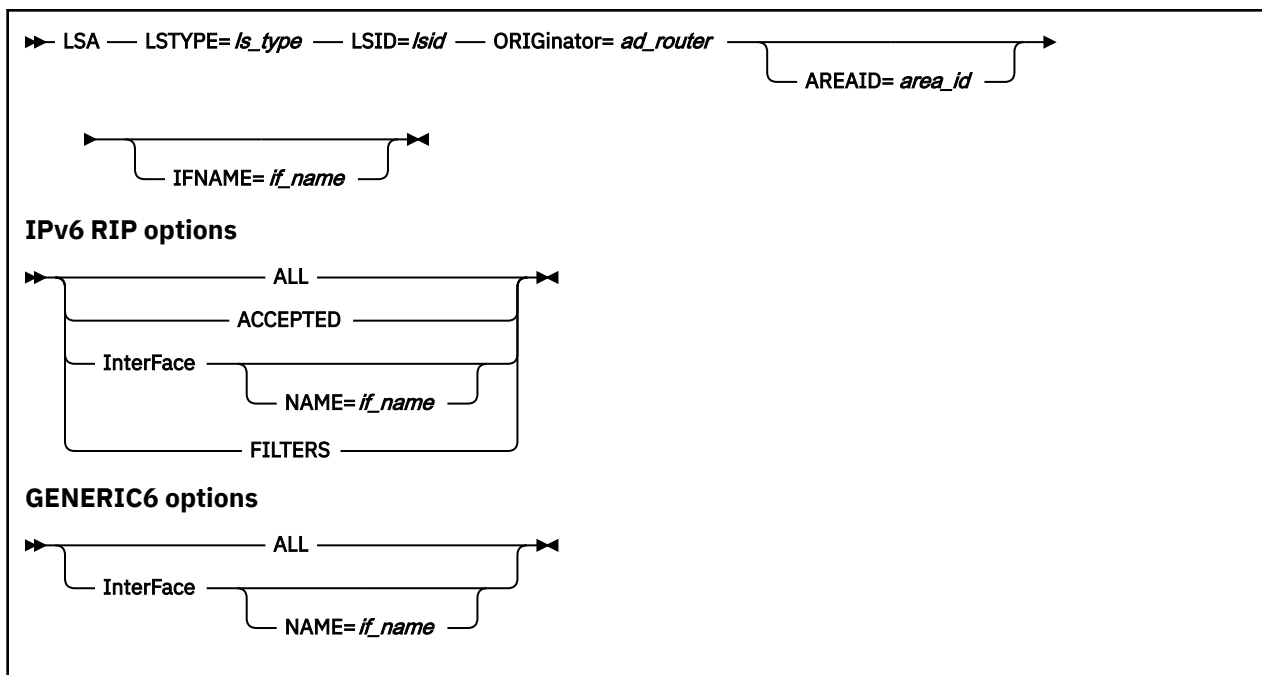
GENERIC options

➤ LIST — ALL — InterFaceS — InterFace —

IPv6 OSPF options

➤ ALL — AREASUM — InterFace — NAME= *if_name* — ACTIVATE — SUSPEND — ID= *if_id* — ACTIVATE — SUSPEND — VLINK — ENDPT= *router-id* — NeighBoR — ID= *router-id* — IFNAME= *if_name* — DBSIZE — IPv6 LSA command — EXTERNAL — DATABASE — AREAID= *area_id* — ROUTERS — STATISTICS — WEIGHT — NAME= *name* — COST= *cost* —

IPv6 LSA command



Purpose

Use the VM Special Message Facility (SMSG) interface to the MPROUTE virtual machine to:

- Stop the MPRoute function
- Reread the MPRoute configuration file
- Start, stop, or change the level of MPRoute tracing for initialization and IPv4 routing protocols
- Set the level of debugging to use for initialization and IPv4 routing protocols
- Start, stop, or change the level of MPRoute tracing for IPv6 routing protocols
- Set the level of debugging to use for IPv6 routing protocols
- Display OSPF information
- Display RIP information
- Display IPv4 information not related to a specific routing protocol
- Display all routes in the MPRoute routing table
- Display IPv6 OSPF information
- Display IPv6 RIP information
- Display IPv6 information not related to a specific dynamic routing protocol
- Display all routes in the MPRoute IPv6 routing table, including deleted routes.

Operands

server_id

Specifies the user ID of the MPRoute server virtual machine.

SHUTDOWN

Stops the MPRoute function.

RECONFIG

Rereads the MPRoute configuration file. This command ignores all statements in the configuration file except new OSPF_Interface, RIP_Interface, Interface, IPv6_RIP_Interface, and IPv6_Interface statements.

Rule: These new configuration statements must be reread from the configuration file through this command before the interface is configured to the TCP/IP stack.

TRACE=trace_level

Starts, stops, or changes the level of MPRoute tracing for initialization and IPv4 routing protocols. The different trace levels available and their descriptions are as follows:

TRACE=0

Turns off MPRoute tracing.

TRACE=1

Gives all the informational messages.

TRACE=2

Gives the informational messages plus formatted packet tracing.



Attention: MPRoute tracing affects MPRoute performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

DEBUG=debug_level

Sets the level of debugging for MPRoute to use for initialization and IPv4 routing protocols.

TRACE6=trace6_level

Starts, stops, or changes the level of MPRoute tracing for IPv6 routing protocols. The different trace levels available and their descriptions are as follows:

TRACE6=0

Turns off MPRoute tracing.

TRACE6=1

Gives all the informational messages.

TRACE6=2

Gives the informational messages plus formatted packet tracing.



Attention: MPRoute tracing affects MPRoute performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

DEBUG6=debug6_level

Sets the level of debugging for MPRoute to use for IPv6 routing protocols.

OSPF

Specifies that OSPF information is to be displayed.

LIST

Specifies that OSPF information is to be displayed as defined in the MPRoute configuration file.

ALL

Displays a comprehensive list of all configuration information.

AREAS

Displays all information concerning configured OSPF areas and their associated ranges.

InterFaceS

Displays, for each OSPF interface, the IP address and configured parameters as coded in the MPRoute configuration file.

NBMA

Displays the interface address and polling interval related to interfaces connected to non-broadcast multi-access networks.

NeighBoRS

Displays the configured neighbors on non-broadcast networks.

VLINKS

Displays all virtual links that have been configured with this router as the endpoint.

LSA

Displays the contents of a single link state advertisement contained in the OSPF database.

A link state advertisement is defined by its

- Link state type (**LSTYPE=ls_type**)

- Link state ID (**LSID=lsid**)
- Advertising router (**ORIGINator=ad_router**).

There is a separate link state database for each OSPF area. **AREAID=area_id** on the command line tells the software which database you want to search. The different kinds of advertisements, which depend on the value given for link-state-type, are:

Router links (LSTYPE=1)

Describes the collected states of a router interface attached to a router.

Network links (LSTYPE=2)

Describes the set of routers attached to a network.

Summary link, IP network (LSTYPE=3)

Describes interarea routes to networks.

Summary link, ASBR (LSTYPE=4)

Describes interarea routes to AS boundary routers.

AS external link (LSTYPE=5)

Describes routes to destinations external to the Autonomous System.

Note: The ORIGINATOR only needs to be specified for link-state-types three, four, and five. The AREAID needs to be specified for all link-state-types except five.

Link State IDs, originators (specified by their router IDs), and area IDs take the same format as IP addresses. For example, the backbone area can be entered as 0.0.0.0.

AREASUM

Displays the statistics and parameters for all OSPF areas attached to the router.

EXTERNAL

Displays the AS external advertisements belonging to the OSPF routing domain. One line is printed for each advertisement.

DATABASE AREAID=area_id

Displays a description of the contents of a particular OSPF area link state database. AS external advertisements are omitted from the display. A single line is printed for each advertisement. If AREAID is not specified, the database from area 0.0.0.0 will be displayed.

DBSIZE

Displays the number of LSAs currently in the link state database, categorized by type.

InterFace NAME=if_name

Displays current, run-time statistics and parameters related to OSPF interfaces. If NAME=*if_name* is omitted, a single line is printed summarizing each interface. If NAME=*if_name* is specified, detailed statistics for that interface will be displayed.

ACTIVATE

Activates an OSPF interface in SUSPEND state to allow adjacency formations with neighbors over this interface. If this is a LAN interface and there is an alternate redundant interface on this same LAN segment that is the primary OSPF interface, this interface becomes a backup interface. ACTIVATE does not force the activated interface to take over or resume the primary OSPF interface role for the LAN segment. This parameter is not applicable for VIPA interfaces.

SUSPEND

Suspends an active OSPF interface that is not in DOWN or SUSPEND state so that adjacency formations with neighbors over this interface are stopped or not allowed. If this is a LAN interface and an alternate interface is available, SUSPEND forces adjacency attempts with neighbors over an alternate redundant interface. Existing connections that use static routes over the suspended interface are not disrupted. If a TCP/IP stack is recycled while an interface is in a suspended state, the interface state is reset after the recycle. This parameter is not applicable for VIPA interfaces.

NeighBoR IPADDR=*ip_addr*

Displays the statistics and parameters related to OSPF neighbors. If IPADDR=*ip_addr* is omitted, a single line is printed summarizing each neighbor. If IPADDR=*ip_addr* is given, detailed statistics for that neighbor are displayed.

ROUTERS

Displays all routes to other routers that have been calculated by OSPF and are currently present in the routing table.

STATISTICS

Displays statistics generated by the OSPF routing protocol. The statistics indicate how well the implementation is performing, including its memory and network utilization. Many of the fields displayed are confirmation of the OSPF configuration.

WEIGHT

Dynamically changes the cost of an OSPF interface. This new cost is flooded quickly throughout the OSPF routing domain, and modifies the routing immediately.

The cost of the interface reverts to its configured value whenever MPRoute is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF interface in the configuration file. This command can only be issued for an OSPF interface that is active in the TCP/IP stack.

NAME=*name*

Is the name of the OSPF interface the new cost affects.

COST=*cost*

Is the new cost value for the OSPF interface.

RIP

Specifies that RIP information is to be displayed.

LIST

Specifies that RIP information is to be displayed as defined in the MPRoute configuration file.

ALL

Displays all RIP-related configuration information.

InterFaceS

Displays IP addresses and configured parameters for each RIP interface.

ACCEPTED

Displays the routes to be unconditionally accepted, as configured with the ACCEPT_RIP_ROUTE statement.

InterFace NAME=*if_name*

Displays statistics and parameters related to RIP interfaces. If NAME=*if_name* is omitted, a single line is printed summarizing each interface. If NAME=*if_name* is given, detailed statistics for the specified interface (*if_name*) are displayed.

FILTERS

Displays the Global RIP filters.

GENERIC

Specifies that IPv4 information not related to a specific routing protocol is to be displayed.

LIST

Specifies that information is to be displayed as defined in the MPRoute configuration file.

ALL

Displays all IPv4 information that is not related to a specific routing protocol.

InterFaceS

Lists all generic IPv4 interfaces that are defined to MPRoute using INTERFACE statements.

InterFace

Displays statistics and parameters related to IPv4 generic interfaces that are known to TCP/IP.

RTTABLE

Displays all of the routes in the MPRoute routing table.

DEST=*ip_addr*

Displays information about a particular route. When multiple equal-cost routes exist, use this command to obtain a list of the next hops.

DELETED

Displays information about routes that were deleted from the MPRoute routing table, and have not been replaced. You cannot use this option with the DEST=*ip_addr* option. If the RIP protocol is running, deleted routes can be displayed for only three minutes after they have been deleted.



Attention: This command displays the contents of the working table that is used by MPRoute, not the TCP/IP routing table. The contents of the MPRoute routing table might contain information different than that in the TCP/IP routing table.

IPv6OSPF

Specifies that IPv6 OSPF information is to be displayed.

ALL

Displays a comprehensive list of IPv6 OSPF information.

AREASUM

Displays the statistics and parameters for all IPv6 OSPF areas attached to the router.

InterFace NAME=*if_name* or InterFace ID=*if_id*

Displays current, run-time statistics and parameters related to IPv6 OSPF interfaces. If NAME=*if_name* and ID=*if_id* are omitted, a single line is printed summarizing each interface. If NAME=*if_name* or ID=*if_id* is specified, detailed statistics for that interface will be displayed.

ACTIVATE

Activates an OSPF interface in SUSPEND state to allow adjacency formations with neighbors over this interface. If this is a LAN interface and there is an alternate redundant interface on this same LAN segment that is the primary OSPF interface, this interface becomes a backup interface. ACTIVATE does not force the activated interface to take over or resume the primary OSPF interface role for the LAN segment. This parameter is not applicable for VIPA interfaces.

SUSPEND

Suspends an active OSPF interface that is not in DOWN or SUSPEND state so that adjacency formations with neighbors over this interface are stopped or not allowed. If this is a LAN interface and an alternate interface is available, SUSPEND forces adjacency attempts with neighbors over an alternate redundant interface. Existing connections that use static routes over the suspended interface are not disrupted. If a TCP/IP stack is recycled while an interface is in a suspended state, the interface state is reset after the recycle. This parameter is not applicable for VIPA interfaces.

VLINK ENDPT=*router-id*

Displays current, run-time statistics and parameters related to IPv6 OSPF virtual links. If ENDPT=*router-id* is omitted, a single line is printed summarizing each virtual link. If ENDPT=*router-id* is specified, detailed statistics for that virtual link will be displayed.

NeighBoR ID=*router-id* IFNAME=*if_name*

Displays the statistics and parameters related to IPv6 OSPF neighbors.

- If ID=*router-id* is omitted, a single line is printed summarizing each neighbor.
- If ID=*router-id* is given, detailed statistics for that neighbor are displayed.
- If the neighbor specified by ID=*router-id* has more than one neighbor relationship with MPRoute (for example if there are multiple IPv6 OSPF links connecting them), IFNAME=*if_name* can be used to specify which link's adjacency to examine (for an adjacency over a virtual link, specify IFNAME=*).

DBSIZE

Displays the number of LSAs currently in the IPv6 OSPF link state database, categorized by type.

LSA

Displays the contents of a single link state advertisement contained in the IPv6 OSPF database. A link state advertisement is defined by its:

- Link state type (LSTYPE=*ls_type*, where *ls_type* is one of the hexadecimal link state type values listed below).
- Link state ID (LSID=*lsid*).
- Advertising router (ORIGinator=*ad_router*).

Each interface has its own set of link LSAs (LSTYPE=0008). IFNAME=*interface_name* on the command line indicates which link's LSA you want to display. There is also a separate link state database for each IPv6 OSPF area. AREAID=*area_id* on the command line indicates which database you want to search. If you do not specify which area to search, the backbone (0.0.0.0) area will be searched. The different kinds of advertisements, which depend on the value given for link state type, are:

Router LSA (LSTYPE=2001)

The complete collection describes the state and cost of the router's interfaces to the area. Each router in an area originates one or more Router LSAs.

Network LSA (LSTYPE=2002)

Originated by the Designated Router of each multiaccess link (i.e., LAN) in the area which supports two or more routers. Describes the set of routers attached to the link, including the Designated Router.

Inter-Area Prefix LSA (LSTYPE=2003)

Originated by an area border router. Describes the route to an IPv6 address prefix that belongs to another area.

Inter-Area Router LSA (LSTYPE=2004)

Originated by an area border router. Describes the route to an AS boundary router that belongs to another area.

AS External LSA (LSTYPE=4005)

Originated by an AS boundary router. Describes the route to a destination external to the IPv6 OSPF Autonomous System.

Link LSA (LSTYPE=0008)

Originated by routers for each link to which they are attached. Provides the router's link-local address, provides a list of IPv6 address prefixes for the link, and asserts a set of options for the Network LSA that will be originated for the link.

Intra-Area Prefix LSA (LSTYPE=2009)

Originated by routers to advertise one or more IPv6 address prefixes that are associated with the router itself, an attached stub network segment, or an attached transit network segment.

Requirements:

1. Specify the AREAID for all link state types except AS External LSA.
Note: The AREAID value defaults to the backbone (0.0.0.0) area if not specified.
2. Specify the IFNAME for Link LSAs (LSTYPE=0008).
3. Originators (specified by their router IDs) and area IDs are specified in dotted-decimal format. For example, the backbone area is entered as 0.0.0.0.

EXTERNAL

Displays the AS external LSAs belonging to the IPv6 OSPF routing domain. One line is printed for each advertisement.

DATABASE AREAID=*area_id*

Displays the contents of a particular IPv6 OSPF area link state database. AS external advertisements are omitted from the display. A single line is printed for each advertisement. If AREAID is not specified, the database from area 0.0.0.0 will be displayed.

ROUTERS

Displays all routes to other routers that have been calculated by IPv6 OSPF and are currently present in the routing table.

STATISTICS

Displays statistics generated by the IPv6 OSPF routing protocol. The statistics indicate how well the implementation is performing, including its memory and network utilization.

WEIGHT

Dynamically changes the cost of an IPv6 OSPF interface. This new cost is flooded quickly throughout the IPv6 OSPF routing domain, and modifies the routing immediately. The cost of the interface reverts to its configured value whenever MPRoute is restarted. To make the cost change permanent, you must reconfigure the appropriate IPv6 OSPF interface in the MPRoute configuration file.

NAME=*name*

Is the name of the IPv6 OSPF interface the new cost affects.

COST=*cost*

Is the new cost value for the IPv6 OSPF interface.

IPv6RIP

Specifies the IPv6 RIP information.

ALL

Displays all IPv6 RIP-related information.

ACCEPTED

Displays the routes to be unconditionally accepted, as configured with the IPV6_ACCEPT_RIP_ROUTE statement.

InterFace NAME=*if_name*

Displays statistics and parameters related to IPv6 RIP interfaces. If NAME=*if_name* is omitted, a single line is printed summarizing each interface. If NAME=*if_name* is given, detailed statistics for the specified interface (*if_name*) are displayed.

FILTERS

Displays the Global IPv6 RIP filters.

GENERIC6

Specifies IPv6 information not related to a specific dynamic routing protocol.

ALL

Displays all IPv6 information not related to a specific routing protocol.

InterFace NAME=*if_name*

Displays statistics and parameters related to IPv6 generic interfaces that are known to TCP/IP or defined to MPRoute with IPV6_INTERFACE statements. If NAME=*if_name* is omitted, a single line is printed summarizing each interface. If NAME=*if_name* is given, detailed statistics for the specified interface (*if_name*) is displayed.

RT6TABLE

Displays all the routes in the MPRoute IPv6 routing table.

DEST=*ip_addr/prefix_len*

Displays information about a particular route. When multiple equal-cost routes exist, use this command to obtain a list of the next hops.

DELETED

Displays information about routes that were deleted from the MPRoute routing table, and have not been replaced. You cannot use this option with the DEST=*ip_addr* option. If the RIP protocol is running, deleted routes can be displayed for only three minutes after they have been deleted.



Attention: This command displays the contents of the working table that is used by MPRoute, not the TCP/IP routing table. The contents of the MPRoute routing table might contain information different than that in the TCP/IP routing table.

OPTIONS

Specifies that the global configuration options information is to be displayed.

HELP**HELPIPV4**

Provides a list of SMSG commands for IPv4 interfaces.

HELPIPV6

Provides a list of SMSG commands for IPv6 interfaces.

Examples

The following are examples of using the SMSG command interface for MPRoute.

All OSPF configuration information: The `msg server_id ospf list all` command lists all OSPF-related configuration information.

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf list all
Ready; T=0.01/0.01 13:02:29
EZZ7831I GLOBAL CONFIGURATION
TRACE: 0, DEBUG: 0
STACK AFFINITY:          TCPIP1
OSPF PROTOCOL:          ENABLED
EXTERNAL COMPARISON:     TYPE 2
AS BOUNDARY CAPABILITY:  DISABLED
DEMAND CIRCUITS:         ENABLED
DR_MAX_ADJ_ATTEMPT:      0

EZZ7832I AREA CONFIGURATION
AREA ID      AUTYPE      STUB?  DEFAULT-COST  IMPORT-SUMMARIES?
1.1.1.1      0=NONE      NO     N/A           N/A
0.0.0.0      0=NONE      NO     N/A           N/A

EZZ7833I INTERFACE CONFIGURATION
IP ADDRESS   AREA      COST  RTRNS  TRDLY  PRI  HELLO  DEAD  DB_EX
10.0.0.17    1.1.1.1    1      5      1      1    10     40    40
10.0.0.21    1.1.1.1    1      5      1      1    10     40    40
10.0.0.14    1.1.1.1    1      5      1      1    10     40    40
10.200.0.1   1.1.1.1    1     N/A    N/A    N/A   N/A     N/A    N/A

DEMAND CIRCUIT PARAMETERS
IP ADDRESS   DONOTAGE  HELLO SUPPRESSION  POLL INTERVAL
10.0.0.17    OFF       ALLOW              60
10.0.0.21    OFF       ALLOW              60
10.0.0.14    OFF       ALLOW              60

SUBNET ADVERTISEMENT PARAMETERS
10.0.0.14

ADVERTISED VIPA ROUTES
10.200.0.0   /255.255.255.0   10.200.0.1   /255.255.255.255

EZZ7836I VIRTUAL LINK CONFIGURATION
VIRTUAL ENDPOINT  TRANSIT AREA  RTRNS  TRNSDLY  HELLO  DEAD  DB_EX
10.0.0.20         1.1.1.1      10      5      30    180   180

EZZ7835I NBMA CONFIGURATION
INTERFACE ADDR    POLL INTERVAL
10.0.0.21        120

EZZ7834I NEIGHBOR CONFIGURATION
NEIGHBOR ADDR    INTERFACE ADDRESS  DR ELIGIBLE?
10.0.0.25        10.0.0.21         NO
10.0.0.23        10.0.0.21         YES
```

TRACE

Displays the level of tracing currently in use by MPRoute for initialization and IPv4 routing protocols.

DEBUG

Displays the level of debugging currently in use by MPRoute for initialization and IPv4 routing protocols.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

OSPF PROTOCOL

Displays that OSPF is enabled or disabled.

EXTERNAL COMPARISON

Displays the external route type used by OSPF when importing external information into the OSPF domain and when comparing OSPF external routes to RIP routes.

AS BOUNDARY CAPABILITY

Indicates whether the router will import external routes into the OSPF domain.

DEMAND CIRCUITS

Indicates whether demand circuit support is available for OSPF interfaces.

DR_MAX_ADJ_ATTEMPT

Specifies a threshold value for the maximum number of adjacency attempts to a neighboring designated router. This value is used for reporting and controlling futile neighbor state loops. See [“Preventing futile neighbor state loops during adjacency formation” on page 198.](#)

The remainder of the `msg server_id ospf list all` output is described in the following sections:

Configured OSPF areas: The `msg server_id ospf list areas` command lists all information concerning configured OSPF areas.

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf list areas
Ready; T=0.01/0.01 13:02:29
EZZ7832I AREA CONFIGURATION
AREA ID      AUTYPE      STUB?  DEFAULT-COST  IMPORT-SUMMARIES?
1.1.1.1      0=NONE      NO      N/A           N/A
0.0.0.0      0=NONE      NO      N/A           N/A
```

AREA ID

Displays the area ID.

AUTYPE

Displays the method used for area authentication. *Simple-pass* means a simple password scheme is being used for the area authentication. *MD5* means MD5 hash is being used for authentication.

STUB?

Indicates whether the area is a stub area.

DEFAULT COST

Displays the cost of the default route configured for the stub area.

IMPORT SUMMARIES?

Indicates whether summary advertisements are to be imported into the stub area.

Note: A stub area that does not allow summaries to be imported is sometimes referred to as a totally stubby area.

Configured OSPF interfaces: The `msg server_id ospf list interfaces` command lists, for each OSPF interface, the IP address and configured parameters as coded in the MPRoute configuration file. (The keyword IFS can be substituted for INTERFACES.)

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf list interfaces
Ready; T=0.01/0.01 13:02:29
EZZ7833I INTERFACE CONFIGURATION
IP ADDRESS    AREA      COST  RTRNS  TRDLY  PRI  HELLO  DEAD  DB_EX
10.0.0.17     1.1.1.1    1      5      1      1    10     40    40
10.0.0.21     1.1.1.1    1      5      1      1    10     40    40
10.0.0.14     1.1.1.1    1      5      1      1    10     40    40
10.200.0.1    1.1.1.1    1     N/A    N/A    N/A   N/A     N/A    N/A

DEMAND CIRCUIT PARAMETERS
IP ADDRESS    DONOTAGE  HELLO SUPPRESSION  POLL INTERVAL
10.0.0.17     OFF       ALLOW              60
```

10.0.0.21	OFF	ALLOW	60
10.0.0.14	OFF	ALLOW	60
SUBNET ADVERTISEMENT PARAMETERS			
10.0.0.14			
ADVERTISED VIPA ROUTES			
10.200.0.0	/255.255.255.0	10.200.0.1	/255.255.255.255

IP ADDRESS

Indicates the IP address of the interface.

AREA

Indicates the OSPF area to which the interface attaches.

COST

Indicates the ToS 0 cost (or metric) associated with the interface.

RTRNS

Indicates the retransmission interval, which is the number of seconds between retransmissions of unacknowledged routing information.

TRDLY

Indicates the transmission delay, which is an estimate of the number of seconds required to transmit routing information over the interface.

PRI

Indicates the interface router priority, which is used when selecting the designated router.

HELLO

Indicates the number of seconds between Hello packets sent from the interface.

DEAD

Indicates the number of seconds after not having received an OSPF Hello packet, that a neighbor is declared to be down.

DB_EX

Indicates the number of seconds to allow the database exchange to complete.

DONOTAGE

Indicates whether the interface is configured as a demand circuit.

HELLO SUPPRESSION

Indicates whether the interface is configured for hello suppression.

POLL INTERVAL

Indicates the interval (in seconds) to be used when attempting to contact a neighbor when a neighbor relationship has failed, but the interface is available.

SUBNET ADVERTISEMENT PARAMETERS

Lists the interfaces that are configured with the Subnet parameter containing a value other than NO. For VIPA interfaces this indicates advertisement of subnet or host routes that are being controlled. For real interfaces this indicates that SUBNET=YES has been coded.

ADVERTISED VIPA ROUTES

Lists the route destinations that MPRoute will advertise for locally owned VIPAs. These advertisements are controlled by the Advertise_VIPA_Routes or Subnet parameter on the OSPF_INTERFACE statement.

Configured OSPF virtual links: The `msg server_id ospf list vlinks` command lists all virtual links that have been configured with this router as an endpoint.

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf list vlinks
Ready; T=0.01/0.01 13:02:29
EZZ7836I VIRTUAL LINK CONFIGURATION
VIRTUAL ENDPOINT      TRANSIT AREA      RTRNS  TRNSDLY  HELLO  DEAD  DB_EX
10.0.0.20              1.1.1.1           10      5       30    180   180
```

VIRTUAL ENDPOINT

Indicates the OSPF router ID of the other endpoint.

TRANSIT AREA

Indicates the non-backbone area through which the virtual link is configured. Virtual links are treated by the OSPF protocol similarly to point-to-point networks.

RTRNS

Indicates the retransmission interval, which is the number of seconds between retransmissions of unacknowledged routing information.

TRNSDLY

Indicates the transmission delay, which is an estimate of the number of seconds required to transmit routing information over the interface.

HELLO

Indicates the number of seconds between Hello packets sent from the interface.

DEAD

Indicates the number of seconds, after not having received an OSPF Hello packet, that a neighbor is declared to be down.

DB_EX

Indicates the number of seconds to allow the database exchange to complete.

Configured OSPF nonbroadcast, multiaccess networks: The `msg server_id ospf list nbma` command lists the interface address and polling interval related to interfaces connected to non-broadcast multi-access networks.

Example: A sample output with explanation of entries follows:

```
msg mproutm1 ospf list nbma
Ready; T=0.01/0.01 13:02:29
EZZ7835I NBMA CONFIGURATION
          INTERFACE ADDR      POLL INTERVAL
          10.0.0.21           120
```

INTERFACE ADDR

Interface IP address.

POLL INTERVAL

Displays the current poll interval value.

Configured OSPF neighbors: The `msg server_id ospf list all neighbors` command lists the configured neighbors on non-broadcast networks. (The keyword NBRS can be substituted for NEIGHBORS.)

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf list all neighbors
Ready; T=0.01/0.01 13:02:29
EZZ7834I NEIGHBOR CONFIGURATION
          NEIGHBOR ADDR      INTERFACE ADDRESS  DR ELIGIBLE?
          10.0.0.25           10.0.0.21         NO
          10.0.0.23           10.0.0.21         YES
```

NEIGHBOR ADDR

Indicates the IP address of the neighbor.

INTERFACE ADDRESS

Indicates the IP address of the interface on which the neighbor is configured.

DR ELIGIBLE?

Indicates whether the neighbor is eligible to become the designated router on the link.

OSPF link state advertisement: The following command displays the contents of a single link state advertisement contained in the OSPF database:

```
msg server_id ospf lsa lstype=ls-type lsid=lsid orig=ad-router areaid=area-id
```

Tips:

1. For a summary of all the non-external advertisements in the OSPF database, use the following command:

```
msg server_id ospf database areaid=area-id
```

2. For a summary of all the external advertisements in the OSPF database, use the following command:

```
msg server_id ospf external
```

Example: The following is an output sample with an explanation of entries:

```
msg mproutm1 ospf lsa lstype=1 lsid=10.0.0.5 orig=10.0.0.5 areaid=1.1.1.1
Ready; T=0.01/0.01 14:19:02
EZZ7880I LSA DETAILS
      LS AGE:          690
      LS OPTIONS:      E,DC (0X22)
      LS TYPE:         1
      LS DESTINATION (ID): 10.0.0.5
      LS ORIGINATOR:   10.0.0.5
      LS SEQUENCE NO:  0X80000006
      LS CHECKSUM:     0X973D
      LS LENGTH:       108
      ROUTER TYPE:     ASBR (0X02)
      # ROUTER IFCS:   1
                  LINK ID:      10.0.0.18
                  LINK DATA:   10.0.0.5
                  INTERFACE TYPE: 1
                        NO. OF METRICS: 0
                        TOS 0 METRIC: 1 (1)
```

LS AGE

Indicates the age of the advertisement in seconds. An asterisk (*) displayed beside the age value indicates that the originator is supporting demand circuits and has indicated that the LSA should not be aged.

LS OPTIONS

Indicates the optional OSPF capabilities supported by the router that originated the advertisement. (The value displayed in parentheses is the hexadecimal options value received in the LSA.) These capabilities are denoted by:

E	Processes type 5 externals; when this is not set, the area to which the advertisement belongs has been configured as a stub.
T	Can route based on ToS.
MC	RFC 1584 (Multicast Extensions to OSPF) is supported. This value is never set by MPRoute but can be received from other routers.
DC	RFC 1793 (Extending OSPF to Support Demand Circuits) is supported.

LS TYPE

Classifies the advertisement and dictates its contents:

1	Router links advertisement
2	Network link advertisement
3	Summary link advertisement
4	Summary ASBR advertisement
5	AS external link

LS DESTINATION

Identifies what is being described by the advertisement. It depends on the advertisement type. For router links and ASBR summaries, it is the OSPF router ID. For network links, it is the IP address of the network designated router. For summary links and AS external links, it is a network or subnet number.

LS ORIGINATOR

OSPF router ID of the originating router.

LS SEQUENCE NO

Used to distinguish separate instances of the same advertisement. Should be looked at as a signed 32-bit integer. Starts at 0x80000001, and increments by 1 each time the advertisement is updated.

LS CHECKSUM

A checksum of advertisement contents, used to detect data corruption.

LS LENGTH

The size of the advertisement in bytes.

ROUTER TYPE

Indicates the level of function of the advertising router. (The value displayed in parentheses is the hexadecimal router type value received in the LSA).

ASBR	The router is an AS boundary router.
ABR	The router is an area border router.
V	The router is an endpoint of an active virtual link that is using the described area as a transit area.

ROUTER IFCS

The number of router interfaces described in the advertisement.

LINK ID

Indicates what the interface connects to. Depends on interface type. For interfaces to routers (that is, point-to-point links), the Link ID is the neighbor router ID. For interfaces to transit networks, it is the IP address of the network designated router. For interfaces to stub networks, it is the network or subnet number.

LINK DATA

Four bytes of extra information concerning the link; it is either the IP address of the interface (for interfaces to point-to-point networks and transit networks), or the subnet mask (for interfaces to stub networks).

INTERFACE TYPE

One of the following:

1	Point-to-point connection to another router
2	Connection to transit network
3	Connection to stub network
4	Virtual link

NO. OF METRICS

The number of nonzero ToS values for which metrics are provided for this interface. For the z/VM implementation, this value will always be 0.

TOS 0 METRIC

The cost of the interface.

The LS age, LS options, LS type, LS destination, LS originator, LS sequence no, LS checksum and LS length fields are common to all advertisements. The Router type and # router ifcs are seen only in router links advertisements. Each link in the router advertisement is described by the Link ID, Link Data, and Interface type fields.

OSPF area statistics and parameters: The `msg server_id ospf areasum` command displays the statistics and parameters for all OSPF areas attached to the router.

Example: A sample output with an explanation of entries follows:

```
smsg mproutm1 ospf areasum
Ready; T=0.01/0.01 14:25:51
EZZ7848I AREA SUMMARY
AREA ID      AUTHENTICATION  #IFCS  #NETS  #RTRS  #BRDRS  DEMAND
1.1.1.1      NONE             4       0       2       0      ON
0.0.0.0      NONE             1       0       0       0      ON
```

AREA ID

Indicates the ID of the area.

AUTHENTICATION

Indicates the default authentication method for the area.

IFCS

Indicates the number of router interfaces attached to the particular area. These interfaces are not necessarily functional.

NETS

Indicates the number of transit networks that have been found while doing the SPF tree calculation for this area.

RTRS

Indicates the number of routers that have been found when doing the SPF tree calculation for this area.

BRDRS

Indicates the number of area border routers that have been found when doing the SPF tree calculation for this area.

DEMAND

Indicates whether demand circuits are supported in this area.

OSPF external advertisements: The `msg server_id ospf external` command lists the AS external advertisements belonging to the OSPF routing domain. One line is printed for each advertisement. Each advertisement is defined by the following three parameters:

- Its link state type (always 5 for AS external advertisements)
- Its link state ID (called the LS destination)
- The advertising router (called the LS originator)

Example: A sample output with an explanation of entries follows:

```
smsg mproutm1 ospf external
Ready; T=0.01/0.01 14:26:28
EZZ7853I AREA LINK STATE DATABASE
TYPE LS DESTINATION  LS ORIGINATOR  SEQNO  AGE  XSUM
5 @10.0.0.2          10.0.0.5      0X80000001 1134  0XFB9A
5 @10.0.3.0          10.0.0.5      0X80000001 1134  0XF89B
# ADVERTISEMENTS:    2
CHECKSUM TOTAL:      0X1F435
```

TYPE

Always 5 for AS external advertisements. An asterisk (*) following the type value indicates that the MC option is on in the advertisement. The MC option indicates that the originating router has implemented RFC 1584 (Multicast Extensions to OSPF). An at sign (@) following the type value indicates that the DC option is on in the advertisement. The DC option indicates that the originating router has implemented RFC 1793 (Extending OSPF to Support Demand Circuits).

LS DESTINATION

Indicates an IP destination (network, subnet, or host). This destination belongs to another autonomous system.

LS ORIGINATOR

Indicates the router that originated the advertisement.

SEQNO, AGE, and XSUM

It is possible for several instances of an advertisement to be present in the OSPF routing domain at any one time. However, only the most recent instance is kept in the OSPF link state database (and printed by this command). The LS sequence number (Seqno), LS age (Age), and LS checksum (Xsum) fields are compared to see which instance is most recent. The LS age field is expressed in seconds. Its maximum value is 3600. An asterisk (*) displayed beside an age value indicates that the DONOTAGE bit is on.

At the end of the display, the total number of AS external advertisements is printed, along with a checksum total over all of their contents. The checksum total is simply the 32-bit sum (carries discarded) of the individual advertisement LS checksum fields. This information can be used to determine quickly whether two OSPF routers have synchronized databases.

OSPF area link state database: The `msg server_id ospf database areaid=area_id` command displays a description of the contents of a particular OSPF area link state database. AS external advertisements are omitted from the display. A single line is printed for each advertisement. Each advertisement is defined by the following three parameters:

- Its link state type (called Type)
- Its link state ID (called the LS destination)
- The advertising router (called the LS originator)

Example: A sample output with an explanation of entries follows:

```
msg mproutm1 ospf database areaid=1.1.1.1
Ready; T=0.01/0.01 14:27:39
EZZ7853I AREA LINK STATE DATABASE
TYPE LS DESTINATION      LS ORIGINATOR      SEQNO      AGE      XSUM
1 @10.0.0.5              10.0.0.5           0X80000006 1206    0X973D
1 @10.0.0.17             10.0.0.17          0X8000002C   5    0X3914
1 @10.0.0.18             10.0.0.18          0X80000049   5    0X5AD8
1 @10.0.0.22             10.0.0.22          0X8000003E 1170    0X7289
1 @10.0.0.125            10.0.0.125         0X80000004  501    0X34D1
2 @10.0.2.1              10.0.0.125         0X80000001  538    0X4AB7
2 @10.0.2.2              10.0.0.18          0X80000023  505    0X2E1C
                        # ADVERTISEMENTS:      7
                        CHECKSUM TOTAL:      0X24B56
```

TYPE

Separate LS types are numerically displayed:

Type 1	Router links advertisements
Type 2	Network links advertisements
Type 3	Network summaries
Type 4	AS boundary router summaries

An asterisk (*) following the type value indicates that the MC option is on in the advertisement. The MC option indicates that the originating router has implemented RFC 1584 (Multicast Extensions to OSPF). An at sign (@) following the type value indicates that the DC option is on in the advertisement. The DC option indicates that the originating router has implemented RFC 1793 (Extending OSPF to Support Demand Circuits).

LS DESTINATION

Indicates what is being described by the advertisement.

LS ORIGINATOR

Indicates the router that originated the advertisement.

SEQNO, AGE, and XSUM

It is possible for several instances of an advertisement to be present in the OSPF routing domain at any one time. However, only the most recent instance is kept in the OSPF link state database (and printed by this command). The LS sequence number (Seqno), LS age (Age) and LS checksum (Xsum) fields are compared to see which instance is most recent. The LS age field is expressed in seconds. Its

maximum value is 3600. An asterisk (*) displayed beside an age value indicates that the DONOTAGE bit is on.

At the end of the display, the total number of advertisements in the area database is printed, along with a checksum total over all of their contents. The checksum total is simply the 32-bit sum (carries discarded) of the individual advertisement LS checksum fields. This information can be used to quickly determine whether two OSPF routers have synchronized databases.

OSPF link state database statistics: The `msg server_id ospf dbsize` command displays the number of LSAs currently in the link state database, categorized by type.

Example: A sample output with explanation of entries follows:

```
smsg mproutm1 ospf dbsize
Ready; T=0.01/0.01 14:28:06
EZZ7854I LINK STATE DATABASE SIZE
# ROUTER-LSAS: 5
# NETWORK-LSAS: 2
# SUMMARY-LSAS: 7
# SUMMARY ROUTER-LSAS: 0
# AS EXTERNAL-LSAS: 2
# INTRA-AREA ROUTES: 5
# INTER-AREA ROUTES: 0
# TYPE 1 EXTERNAL ROUTES: 0
# TYPE 2 EXTERNAL ROUTES: 0
```

OSPF interface statistics and parameters: The `msg server_id ospf interface name=if-name` command displays current, run-time statistics and parameters related to OSPF interfaces. (The keyword IF can be substituted for INTERFACE.) If no NAME= parameter is given (see Example 1), a single line is printed summarizing each interface. If NAME= parameter is given (see Example 2), detailed statistics for that interface will be displayed.

Examples: Sample outputs with explanations of entries follows:

```
1. smsg mproutm1 ospf interface
Ready; T=0.01/0.01 14:29:04
EZZ7849I INTERFACES
IFC ADDRESS    PHYS      ASSOC. AREA  TYPE  STATE  #NBRS  #ADJS
10.0.0.17      M1TOM3    1.1.1.1     P-P   16     1       1
10.0.0.21      M1TOM2    1.1.1.1     P-P   1       2       0
10.0.0.14      M1TOM0    1.1.1.1     P-P   16     0       0
10.200.0.1     LVIPA1    1.1.1.1     VIPA   N/A    N/A     N/A
UNNUMBERED     VL/0      0.0.0.0     VLINK  1       1       0
```

IFC ADDRESS

Interface IP address.

PHYS

Displays the interface name.

ASSOC AREA

Attached area ID.

TYPE

Interface type. Can be BRDCST (a broadcast interface), P-P (a point-to-point interface), MULTI (a non-broadcast, multiaccess interface), VLINK (an OSPF virtual link), or VIPA (a Virtual IP Address link).

STATE

Can be one of the following values. With the exception of Suspend, these values are described in RFC 1583 (OSPF Version 2).

1*	Suspend - The interface is suspended because an SMSG command was issued or because it was unable to establish an adjacency with a neighboring designated router after it exceeded the futile neighbor state loop threshold DR_Max_Adj_Attempt). See “Preventing futile neighbor state loops during adjacency formation” on page 198 .
1	Down

2	Backup
4	Looped back
8	Waiting
16	Point-to-point
32	DR other
64	Backup DR
128	Designated router

#NBRS

Number of neighbors. This is the number of routers whose hellos have been received, plus those that have been configured.

#ADJS

Number of adjacencies. This is the number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.

```
2. smsg mproutm1 ospf interface name=m1tom3
Ready; T=0.01/0.01 14:29:32
EZZ7850I INTERFACE DETAILS
                INTERFACE ADDRESS:    10.0.0.17
                ATTACHED AREA:        1.1.1.1
                PHYSICAL INTERFACE:    M1TOM3
                INTERFACE MASK:        255.255.255.252
                INTERFACE TYPE:        P-P
                STATE:                 16
                DESIGNATED ROUTER:     N/A
                BACKUP DR:             N/A

DR PRIORITY:    N/A  HELLO INTERVAL:    10  RXMT INTERVAL:    5
DEAD INTERVAL:    40  TX DELAY:          1  POLL INTERVAL:    N/A
DEMAND CIRCUIT:  OFF  HELLO SUPPRESS:  OFF  SUPPRESS REQ:    OFF
MAX PKT SIZE:    32696  TOS 0 COST:      1  DB_EX INTERVAL:    40
AUTH TYPE:      NONE

# NEIGHBORS:      1  # ADJACENCIES:    1  # FULL ADJS.:      1
# MCAST FLOODS:  44  # MCAST ACKS:    20  # MAX ADJ. RESETS: N/A

NETWORK CAPABILITIES:
POINT-TO-POINT
DEMAND-CIRCUITS
MULTICAST
```

INTERFACE ADDRESS

Interface IP address.

ATTACHED AREA

Attached area ID.

PHYSICAL INTERFACE

Displays interface name.

INTERFACE MASK

Displays interface subnet mask.

INTERFACE TYPE

Can be BRDCST (a broadcast interface), P-P (a point-to-point interface), MULTI (a non-broadcast, multiaccess interface), VLINK (an OSPF virtual link), or VIPA (a Virtual IP Address link).

STATE

Can be one of the following values. With the exception of Suspend, these values are described in RFC 1583 (OSPF Version 2).

1*	Suspend - The interface is suspended because an SMSG command was issued or because it was unable to establish an adjacency with a neighboring designated router after it exceeded the futile neighbor state loop threshold (DR_Max_Adj_Attempt). See “Preventing futile neighbor state loops during adjacency formation” on page 198.
1	Down
2	Backup
4	Looped back
8	Waiting
16	Point-to-point
32	DR other
64	Backup DR
128	Designated router

DESIGNATED ROUTER

IP address of the designated router.

BACKUP DR

IP address of the backup designated router.

DR PRIORITY

Displays the interface router priority used when selecting the designated router. A higher value indicates that this MPRoute is more likely to become the designated router. A value of 0 indicates that MPRoute will never become the designated router.

HELLO INTERVAL

Displays the current hello interval value.

RXMT INTERVAL

Displays the current retransmission interval value.

DEAD INTERVAL

Displays the current dead interval value.

TX DELAY

Displays the current transmission delay value.

POLL INTERVAL

Displays the current poll interval value.

DEMAND CIRCUIT

Displays the current demand circuit status.

HELLO SUPPRESS

Displays whether Hello Suppression is currently on or off.

SUPPRESS REQ

Displays whether Hello Suppression was requested.

MAX PKT SIZE

Displays the maximum size for an OSPF packet sent out this interface.

TOS 0 COST

Displays the interface ToS 0 cost.

DB_EX INTERVAL

Indicates the number of seconds to allow the database exchange to complete.

AUTH TYPE

Authentication type is one of the following:

NONE

No authentication is used.

Password

Simple password authentication.

MD5

Crypto-MD5 type authentication.

NEIGHBORS

Number of neighbors. This is the number of routers whose hellos have been received, plus those that have been configured.

ADJACENCIES

Number of adjacencies. This is the number of neighbors in state Exchange or greater.

FULL ADJS

Number of full adjacencies. This is the number of neighbors whose state is Full (and therefore with which the router has synchronized databases).

MAX ADJ. RESETS

Total number of times the maximum threshold value for attempting an adjacency with a neighboring designated router has been reset. The value N/A indicates that the field is not applicable for that interface, based on the interface type that is used to reach a neighbor. For more information, see the description of DR_MAX_ADJ_ATTEMPT in the [“Examples” on page 271](#) section (All OSPF configuration information).

MCAST FLOODS

Number of link state updates that flooded the interface (not counting retransmissions).

MCAST ACKS

Number of link state acknowledgments that flooded the interface (not counting retransmissions).

NETWORK CAPABILITIES

Displays the capabilities of the interface.

OSPF neighbor statistics and parameters: The `msg server_id ospf neighbor ipaddr=ip-addr` command displays the statistics and parameters related to OSPF neighbors. (The keyword NBR can be substituted for NEIGHBOR.) If no IPADDR= parameter is given (see Example 1), a single line is printed summarizing each neighbor. If an IPADDR= parameter is given (see Example 2), detailed statistics for that neighbor are displayed.

Examples: Sample outputs with explanation of entries follows:

```
1. msg mproutm1 ospf neighbor
Ready; T=0.01/0.01 14:30:41
EZZ7851I NEIGHBOR SUMMARY
NEIGHBOR ADDR  NEIGHBOR ID      STATE  LSRXL  DBSUM  LSREQ  HSUP  IFC
10.0.0.18      10.0.0.18      128    0      0      0      OFF  M1TOM3
10.0.0.25      0.0.0.0        1      0      0      0      OFF  M1TOM2
10.0.0.23      0.0.0.0        1      0      0      0      OFF  M1TOM2
VL/0           10.0.0.20      1      0      0      0      OFF  *
```

NEIGHBOR ADDR

Displays the neighbor interface IP address.

NEIGHBOR ID

Displays the neighbor OSPF router ID.

STATE

Can be one of the following:

1	Down
2	Attempt
4	Init
8	2-Way

16	ExStart
32	Exchange
64	Loading
128	Full

For more information about these values, refer to RFC 1583 (OSPF Version 2).

LSRXL

Displays the size of the current link state retransmission list for this neighbor.

DBSUM

Displays the size of the database summary list waiting to be sent to the neighbor.

LSREQ

Displays the number of link state advertisements that are being requested from the neighbor.

HSUP

Displays whether Hello Suppression is active with the neighbor.

IFC

Displays the name of the interface over which a relationship has been established with this neighbor.

```
2. msg mproutm1 ospf neighbor ipaddr=10.0.0.18
Ready; T=0.01/0.01 14:30:57
EZZ7852I NEIGHBOR DETAILS
      NEIGHBOR IP ADDRESS:    10.0.0.18
      OSPF ROUTER ID:        10.0.0.18
      NEIGHBOR STATE:        128
      PHYSICAL INTERFACE:    M1TOM3
      DR CHOICE:              0.0.0.0
      BACKUP CHOICE:          0.0.0.0
      DR PRIORITY:            1
      NBR OPTIONS:            E (0X02)
DB SUMM QLEN: 0  LS RXMT QLEN: 0  LS REQ QLEN: 0
LAST HELLO: 7  NO HELLO: OFF
# LS RXMITS: 0  # DIRECT ACKS: 0  # DUP LS RCVD: 0
# OLD LS RCVD: 0  # DUP ACKS RCVD: 1  # NBR LOSSES: 0
# ADJ. RESETS: 0
```

NEIGHBOR IP ADDRESS

Displays the neighbor interface IP address.

OSPF ROUTER ID

Neighbor OSPF router ID.

NEIGHBOR STATE

Can be one of the following:

- 1 (Down)
- 2 (Attempt)
- 4 (Init)
- 8 (2-Way)
- 16 (ExStart)
- 32 (Exchange)
- 64 (Loading)
- 128 (Full)

PHYSICAL INTERFACE

Displays the name of the interface over which a relationship has been established with this neighbor.

DR CHOICE, BACKUP CHOICE, DR PRIORITY

Indicates the values seen in the last hello received from the neighbor.

NBR OPTIONS

Indicates the optional OSPF capabilities supported by the neighbor. (The value displayed in parentheses is the hexadecimal options value received from the neighbor). These capabilities are denoted by:

- E (processes type 5 externals; when this is not set, the area to which the common network belongs has been configured as a stub)
- T (can route based on ToS)
- MC (can forward IP multicast datagrams)
- DC (can support demand circuits)

This field is valid only for those neighbors in state Exchange or greater.

DB SUMM QLEN

Indicates the number of advertisements waiting to be summarized in Database Description packets. It should be 0 except when the neighbor is in state Exchange.

LS RXMT QLEN

Indicates the number of advertisements that have been flooded to the neighbor, but not yet acknowledged.

LS REQ QLEN

Indicates the number of advertisements that are being requested from the neighbor in state Loading.

LAST HELLO

Indicates the number of seconds since a hello has been received from the neighbor.

NO HELLO

Indicates whether Hello Suppression is active with the neighbor.

LS RXMITS

Indicates the number of retransmissions that have occurred during flooding.

DIRECT ACKS

Indicates responses to duplicate link state advertisements.

DUP LS RCVD

Indicates the number of duplicate retransmissions that have occurred during flooding.

OLD LS RCVD

Indicates the number of old advertisements received during flooding.

DUP ACKS RCVD

Indicates the number of duplicate acknowledgments received.

NBR LOSSES

Indicates the number of times the neighbor has transitioned to Down state.

ADJ. RESETS

Counts transitions to state ExStart from a higher state.

OSPF router routes: The `msg server_id ospf routers` command displays all routes to other area-border or autonomous system boundary routers that have been calculated by OSPF and are now present in the routing table.

Example: A sample output with explanations of entries follows:

```
msg mproutm1 ospf routers
Ready; T=0.01/0.01 08:30:10
EZZ7855I OSPF ROUTERS
DTYPE RTYPE DESTINATION AREA COST NEXT HOP(S)
ASBR SPF 10.0.0.5 1.1.1.1 1 10.0.0.13 *
```

DTYPE

Indicates the destination type:

ASBR

Indicates that the destination is an AS boundary router.

ABR

Indicates that the destination is an area border router.

FADD

Indicates a forwarding address (for external routes).

RTYPE

Indicates the route type and how the route was derived:

SPF

Indicates that the route is an intra-area route (comes from the Dijkstra calculation).

SPIA

Indicates that it is an inter-area route (comes from considering summary link advertisements).

DESTINATION

Indicates the destination router OSPF router ID.

AREA

Displays the OSPF area to which the destination router belongs.

COST

Displays the cost to reach the router.

NEXT HOP(S)

Indicates the address of the next router on the path toward the destination host. A number in parentheses at the end of the column indicates the number of equal-cost routes to the destination.

OSPF routing protocol statistics: The `msg server_id ospf statistics` command displays statistics generated by the OSPF routing protocol. (The keyword `STATS` can be substituted for `STATISTICS`.) The statistics indicate how well the implementation is performing, including its memory and network utilization. Many of the fields displayed are confirmation of the OSPF configuration.

Example: A sample output with explanation of entries follows:

```
msg mproutm0 ospf statistics
Ready; T=0.01/0.01 10:10:00
EZZ7856I OSPF STATISTICS
      OSPF ROUTER ID:      10.0.0.5 (*ROUTERID)
      EXTERNAL COMPARISON:  TYPE 2
      AS BOUNDARY CAPABILITY: YES
      IMPORT EXTERNAL ROUTES: RIP SUB
      ORIG. DEFAULT ROUTE:  ALWAYS
      DEFAULT ROUTE COST:   (1, TYPE 2)
      DEFAULT FORWARD. ADDR.: 10.1.1.2
      LEARN HIGHER COST DFLT: NO

ATTACHED AREAS:          1  OSPF PACKETS RCVD:          5
OSPF PACKETS RCVD W/ERRS: 0  TRANSIT NODES ALLOCATED:      8
TRANSIT NODES FREED:      6  LS ADV. ALLOCATED:          9
LS ADV. FREED:            6  QUEUE HEADERS ALLOC:       32
QUEUE HEADERS AVAIL:      29 MAXIMUM LSA SIZE:         32696
# DIJKSTRA RUNS:          1  INCREMENTAL SUMM. UPDATES:    0
INCREMENTAL VL UPDATES:   0  MULTICAST PKTS SENT:         8
UNICAST PKTS SENT:        4  LS ADV. AGED OUT:            0
LS ADV. FLUSHED:          0  PTRS TO INVALID LS ADV:      0
INCREMENTAL EXT. UPDATES:  1
```

OSPF ROUTER ID

Displays the router OSPF router ID.

EXTERNAL COMPARISON

Displays the external route type used by OSPF when importing external information into the OSPF domain and when comparing OSPF external routes to RIP routes.

AS BOUNDARY CAPABILITY

Displays whether external routes will be imported.

IMPORT EXTERNAL ROUTES

Displays the external routes that will be imported. Displayed only when AS Boundary Capability is enabled.

ORIG. DEFAULT ROUTE

Displays whether the router will advertise an OSPF default route. Displayed only when AS Boundary Capability is enabled.

DEFAULT ROUTE COST

Displays the cost and type of the default route (if advertised). Displayed only when AS Boundary Capability is enabled and Orig Default Route is ALWAYS.

DEFAULT FORWARD ADDR

Displays the forwarding address specified in the default route (if advertised). Displayed only when AS Boundary Capability is enabled and Orig Default Route is ALWAYS.

LEARN HIGHER COST DFLT

Indicates the value of the LEARN_DEFAULT_ROUTE parameter of the AS_BOUNDARY_ROUTING configuration statement. Displayed only when AS Boundary Capability is enabled and Orig Default Route is ALWAYS.

ATTACHED AREAS

Indicates the number of areas that the router has active interfaces to.

OSPF PACKETS RCVD

Covers all types of OSPF protocol packets.

OSPF PACKETS RCVD W/ERRS

Indicates the number of OSPF packets that have been received that were determined to contain errors.

TRANSIT NODES

Allocated to store router links and network links advertisements.

LS ADV

Allocated to store summary link and AS external link advertisements.

QUEUE HEADERS

Form lists of link state advertisements. These lists are used in the flooding and database exchange processes; if the number of queue headers allocated is not equal to the number available, database synchronization with a neighbor is in progress.

MAXIMUM LSA SIZE

The size of the largest link state advertisement that can be sent.

DIJKSTRA RUNS

Indicates how many times the OSPF routing table has been calculated from scratch.

INCREMENTAL SUMM UPDATES, INCREMENTAL VL UPDATES

Indicates that new summary link advertisements have caused the routing table to be partially rebuilt.

MULTICAST PKTS SENT

Covers OSPF hello packets and packets sent during the flooding procedure.

UNICAST PKTS SENT

Covers OSPF packet retransmissions and the Database Exchange procedure.

LS ADV. AGED OUT

Indicates the number of advertisements that have hit 60 minutes. Link state advertisements are aged out after 60 minutes. Usually they are refreshed before this time.

LS ADV. FLUSHED

Indicates the number of advertisements removed (and not replaced) from the link state database.

PTRS TO INVALID LS ADV

Indicates the number of pointers on the LSA retransmit queue that point to invalid LS advertisements.

INCREMENTAL EXT. UPDATES

Displays the number of changes to external destinations that are incrementally installed in the routing table.

The following are examples of using the SMSG RIP command.

All RIP configuration information: The `msg server_id rip list all` command lists all RIP-related configuration information.

Example: A sample output with explanations of entries follows:

```
smsg mproutm4 rip list all
Ready; T=0.01/0.01 08:56:06
EZZ7843I RIP CONFIGURATION
TRACE: 0, DEBUG: 0
STACK AFFINITY: TCIPM4
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
M4TOM0          10.0.0.2      RIP-2 MULTICAST
                                SEND NET AND SUBNET ROUTES
                                RECEIVE NO DYNAMIC HOST ROUTES
                                RIP INTERFACE INPUT METRIC: 1
                                RIP INTERFACE OUTPUT METRIC: 0
                                RIP RECEIVE CONTROL: RIP2
M4TOGLAN3       10.0.3.1      RIP-2 MULTICAST
                                SEND NET AND SUBNET ROUTES
                                RECEIVE NO DYNAMIC HOST ROUTES
                                RIP INTERFACE INPUT METRIC: 1
                                RIP INTERFACE OUTPUT METRIC: 0
                                RIP RECEIVE CONTROL: RIP2

EZZ7844I RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  10.2.0.1
IGNORE RIP UPDATES FROM:
  NONE
```

TRACE

Displays the level of tracing currently in use by MPRoute for initialization and IPv4 routing protocols.

DEBUG

Displays the level of debugging currently in use by MPRoute for initialization and IPv4 routing protocols.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

The remainder of the `smsg server_id rip list all` output is described in the following sections.

Configured RIP interfaces: The `smsg server_id rip interface` command lists IP addresses and configured parameters for each RIP interface. (The keyword `IFS` can be substituted for `INTERFACES`.)

Example: A sample command and output with explanations follows:

```
smsg mproutm4 rip interface
Ready; T=0.01/0.01 08:56:06
EZZ7843I RIP CONFIGURATION
TRACE: 0, DEBUG: 0
STACK AFFINITY: TCIPM4
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
M4TOM0          10.0.0.2      RIP-2 MULTICAST
                                SEND NET AND SUBNET ROUTES
                                RECEIVE NO DYNAMIC HOST ROUTES
                                RIP INTERFACE INPUT METRIC: 1
                                RIP INTERFACE OUTPUT METRIC: 0
                                RIP RECEIVE CONTROL: RIP2
M4TOGLAN3       10.0.3.1      RIP-2 MULTICAST
                                SEND NET AND SUBNET ROUTES
                                RECEIVE NO DYNAMIC HOST ROUTES
                                RIP INTERFACE INPUT METRIC: 1
                                RIP INTERFACE OUTPUT METRIC: 0
                                RIP RECEIVE CONTROL: RIP2
```

RIP

Indicates whether RIP communication is enabled.

RIP DEFAULT ORIGINATION

Indicates the conditions under which RIP supports default route generation and the advertised cost for the default route.

PER-INTERFACE ADDRESS FLAGS

Specifies information about an interface:

RIP VERSION

Specifies whether RIP Version 1 or RIP Version 2 packets are being sent over this interface.

SEND

Specifies which types of routes will be included in RIP responses sent out on this interface.

RECEIVE

Specifies which types of routes will be accepted in RIP responses received on this interface.

RIP INTERFACE INPUT METRIC

Specifies the value of the metric to be added to RIP routes received over this interface.

RIP INTERFACE OUTPUT METRIC

Specifies the value of the metric to be added to RIP routes advertised over this interface.

RIP RECEIVE CONTROL

Indicates what level of RIP updates can be received over the interface. Values are:

ANY

RIP1 and RIP2 updates can be received.

NO

No RIP updates can be received.

RIP1

Only RIP1 updates can be received.

RIP2

Only RIP2 updates can be received.

RIP routes to be accepted: The `msg server_id rip list accepted` command lists the routes to be unconditionally accepted, as configured with the `ACCEPT_RIP_ROUTE` statement.

Example: A sample output with explanation of entries follows:

```
msg mproutm4 list accepted
Ready; T=0.01/0.01 08:56:06
EZZ7844I RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  10.2.0.1
IGNORE RIP UPDATES FROM:
  NONE
```

ACCEPT RIP UPDATES ALWAYS FOR

Indicates the networks, subnets, and hosts for which updates are always accepted.

RIP interface statistics and parameters: The `msg server_id rip interface name=if-name` command displays statistics and parameters related to RIP interfaces. (The keyword `IF` can be substituted for `INTERFACE`.) If no `NAME=` parameter is given (`msg server_id rip interface`), a single line is printed summarizing each interface. (See Example 1.) If a `NAME=` parameter is given, detailed statistics for that interface are displayed. (See Example 2.)

Examples:

```
1. msg mproutm4 rip interface
Ready; T=0.01/0.01 08:59:19
EZZ7859I RIP INTERFACES
IFC ADDRESS      IFC NAME      SUBNET MASK    MTU    DESTINATION
10.0.0.2         M4TOM0        255.255.255.252 32760  10.0.0.1
10.0.3.1         M4TOGLAN3     255.255.255.0  8192   N/A
```

IFC ADDRESS

Indicates the interface IP address.

IFC NAME

Indicates the interface name.

SUBNET MASK

Indicates the subnet mask.

MTU

Indicates the value of the maximum transmission unit.

DESTINATION

Indicates the RIP identification for the destination router when the interface is point-to-point.

```
2. smsg mproutm4 rip interface name=m4tom0
Ready; T=0.01/0.01 10:28:51
EZZ7860I RIP INTERFACE DETAILS
INTERFACE ADDRESS:    10.0.0.2
INTERFACE NAME:       M4TOM0
SUBNET MASK:          255.255.255.252
MTU                   32760
DESTINATION ADDRESS:  10.0.0.1

RIP VERSION:          2      SEND POIS. REV. ROUTES: YES
IN METRIC:             1      OUT METRIC:             0
RECEIVE NET ROUTES:    YES    RECEIVE SUBNET ROUTES: YES
RECEIVE HOST ROUTES:   NO     SEND DEFAULT ROUTES:   NO
SEND NET ROUTES:       YES    SEND SUBNET ROUTES:   YES
SEND STATIC ROUTES:    NO     SEND HOST ROUTES:    NO

SEND ONLY: ALL

FILTERS: NOSEND        10.0.2.2          255.255.255.255

RIP RECEIVE CONTROL:   RIP2
```

INTERFACE ADDRESS

Indicates the interface IP address.

INTERFACE NAME

Indicates the interface name.

SUBNET MASK

Indicates the subnet mask.

MTU

Indicates the value of the maximum transmission unit.

DESTINATION ADDRESS

Indicates the RIP identification for the destination router when the interface is point-to-point.

RIP VERSION

Indicates whether RIP Version 1 or RIP Version 2 packets are sent over this interface.

SEND POIS. REV. ROUTES

Indicates whether poisoned reverse routes are advertised in RIP responses sent over this interface. A poisoned reverse route is one with an infinite metric (a metric of 16).

IN METRIC

Specifies the value of the metric to be added to RIP routes received over this interface.

OUT METRIC

Specifies the value of the metric to be added to RIP routes advertised over this interface.

RECEIVE NET ROUTES

Indicates whether network routes are accepted in RIP responses received over this interface.

RECEIVE SUBNET ROUTES

Indicates whether subnet routes are accepted in RIP responses received over this interface.

RECEIVE HOST ROUTES

Indicates whether host routes are accepted in RIP responses received over this interface.

SEND DEFAULT ROUTES

Indicates whether the default route, if available, is advertised in RIP responses sent over this interface.

SEND NET ROUTES

Indicates whether network routes are advertised in RIP responses sent over this interface.

SEND SUBNET ROUTES

Indicates whether subnet routes are advertised in RIP responses sent over this interface.

SEND STATIC ROUTES

Indicates whether static routes are advertised in RIP responses sent over this interface.

SEND HOST ROUTES

Indicates whether host routes are advertised in RIP responses sent over this interface.

SEND ONLY

Indicates the route-type restrictions on RIP broadcasts for this interface.

FILTERS

Indicates the send and receive filters for this interface.

RIP RECEIVE CONTROL

Indicates the type of RIP packets that will be received over this interface: RIP1, RIP2, ANY (both RIP1 and RIP2), or NONE.

Global RIP filters: The `msg server_id rip filters` command displays the Global RIP filters.

Example: A sample output with explanations of entries follows.

```
msg mproutm4 rip filters
Ready; T=0.01/0.01 09:06:31
EZZ8016I GLOBAL RIP FILTERS
SEND ONLY: VIRTUAL, DEFAULT

IGNORE RIP UPDATES FROM:
 10.0.3.4

FILTERS: NOSEND          10.5.0.1          255.255.255.255
RIP RECEIVE CONTROL:    RIP2
```

SEND ONLY

Indicates the global route-type restrictions on RIP broadcasts that apply to all RIP interfaces.

IGNORE RIP UPDATES FROM

Specifies that RIP routing table broadcasts from this gateway are to be ignored. This option serves as a RIP input filter.

FILTERS

Indicates the global send and receive filters that apply to all RIP interfaces.

The following are examples of using the MSG GENERIC command.

All IPv4 generic information: The `msg server_id generic list all` command lists all IPv4 configuration information that is not related to a specific routing protocol.

Example: A sample output with explanations of entries follows:

```
msg mproutm4 generic list all
Ready; T=0.01/0.01 09:21:14
EZZ8053I IPV4 GENERIC CONFIGURATION
TRACE: 0, DEBUG: 0
IPV4 TRACE DESTINATION: N/A
STACK AFFINITY: TCP4PM4

EZZ8056I IPV4 GEN INT CONFIGURATION
IFC NAME      IFC ADDRESS      SUBNET MASK      MTU DESTADDR
M4TOM7        10.0.7.2          255.255.255.252 32760 10.0.7.1
```

TRACE

Displays the level of tracing currently in use by MPRoute initialization and IPv4 routing protocols.

DEBUG

Displays the level of debugging currently in use by MPRoute initialization and IPv4 routing protocols.

IPV4 TRACE DESTINATION

Indicates the file name of the destination for IPv4 trace, or OMPCTRC if the destination is the MPRoute CTRACE.

Restriction: On the console, the file name is shown in upper case, regardless of the case of the actual file name.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

IPv4 GENERIC INT CONFIGURATION

Displays the same output as `msg server_id generic list interfaces` described below in Configured IPv4 generic interfaces.

Configured IPv4 generic interfaces: The `msg server_id generic list interfaces` command lists, for each IPv4 generic interface, the IP address and configured parameters that are defined to MPRoute using the INTERFACE statement. IFS can be used in place of INTERFACES.

Example: A sample output with explanations of entries follows:

```
msg mproutm4 generic list interfaces
Ready; T=0.01/0.01 09:21:14
EZZ8056I IPV4 GEN INT CONFIGURATION
IFC NAME      IFC ADDRESS      SUBNET MASK      MTU DESTADDR
M4TOM7        10.0.7.2          255.255.255.252 32760 10.0.7.1
```

IFC NAME

The interface link name, as defined using the NAME parameter on the INTERFACE statement.

IFC ADDRESS

The interface home address, as defined using the IP_ADDRESS parameter on the INTERFACE statement.

SUBNET MASK

The interface subnet mask, as defined using the SUBNET_MASK parameter on the INTERFACE statement.

MTU

The interface MTU size, as defined using the MTU parameter on the INTERFACE statement.

DESTADDR

If the interface is known to be a point-to-point interface and the DESTINATION_ADDR parameter was coded in the MPRoute configuration file, DESTADDR is the value of the interface DESTINATION_ADDR parameter. Otherwise, N/A is displayed.

IPv4 generic interfaces: The `msg server_id generic interface` command displays current, run-time statistics and parameters related to IPv4 generic interfaces that are known to TCP/IP. The keyword IF can be used instead of INTERFACE.

Example: A sample output with explanations of entries follows:

```
msg mproutm4 generic interface
Ready; T=0.01/0.01 09:26:11
EZZ8060I IPV4 GENERIC INTERFACES
IFC NAME      IFC ADDRESS      SUBNET MASK      MTU  CFG  IGN
M4TOM5        10.0.5.2          N/A              N/A   NO   YES
M4TOM7        10.0.7.2          255.255.255.252 32760 YES   NO
```

IFC NAME

The interface link name.

IFC ADDRESS

The interface home address.

SUBNET MASK

The interface subnet mask. If the interface is being ignored by MPRoute, N/A is displayed.

MTU

The interface MTU size. If the interface is being ignored by MPRoute, N/A is displayed.

CFG

Indicates whether or not the interface was configured to MPRoute.

IGN

Indicates whether or not the interface is being ignored by MPRoute (this field can only be YES if CFG=NO, and GLOBAL_OPTIONS IGNORE_UNDEFINED_INTERFACES is configured to be YES.)

The following are examples of using the SMSG RTTABLE command

MPRoute IPv4 routing table: The `msg server_id rtttable` command displays all of the routes in the MPRoute IPv4 routing table.



Attention: This command displays the contents of the working table that is used by MPRoute, not the TCP/IP routing table. The contents of the MPRoute routing table might contain different information than that in the TCP/IP routing table.

Example: A sample output with explanation of entries follows:

```
msg mproutm0 rtttable
Ready; T=0.01/0.01 09:29:46
EZZ7847I ROUTING TABLE
TYPE    DEST NET      MASK      COST      AGE      NEXT HOP(S)
SBNT    10.0.0.0        FF000000  1         19       NONE
DIR*    10.0.0.0        FFFFFFFF  1         47       10.0.0.1
RIP     10.0.0.2        FFFFFFFF  1         10       10.0.0.2
DIR*    10.0.0.8        FFFFFFFF  1         60856    10.0.0.9
SPF     10.0.0.9        FFFFFFFF  2         60852    M0TOM2
SPF     10.0.0.10       FFFFFFFF  1         60852    10.0.0.10
DIR*    10.0.0.12       FFFFFFFF  1         60856    10.0.0.13
SPF*    10.0.1.0        FFFFFFF0  1         60856    M0TOGLAN1
SPF     10.0.2.0        FFFFFFF0  2         60852    10.0.0.6      (2)
RIP     10.0.3.0        FFFFFFF0  2         10       10.0.0.2
0 NETS DELETED, 1 NETS INACTIVE
```

TYPE

Indicates how the route was derived:

DFLT

Indicates a route defined using the `DEFAULT_ROUTE` configuration statement in the MPRoute configuration file.

SBNT

Indicates that the network is subnetted; such an entry is a placeholder only.

DIR

Indicates a directly connected network, subnet, or host.

RIP

Indicates a route that was learned through the RIP protocol.

DEL

Indicates the route has been deleted.

Restriction: Deleted routes are shown in this display only if RIP is active and only as long as RIP needs to advertise to neighboring routers that they have been deleted. Deleted routes cannot be displayed in the detailed routes display.

STAT

Indicates a nonreplaceable statically configured route.

SPF

Indicates that the route is an OSPF intra-area route.

SPIA

Indicates that the route is an OSPF interarea route.

SPE1

Indicates OSPF external routes (type 1).

SPE2

Indicates OSPF external routes (type 2)

RNGE

Indicates a route type that is an active OSPF area address range and is not used in forwarding packets.

RSTA

Indicates a static route that is defined as replaceable.

An asterisk (*) after the route type indicates that the route has a directly connected backup. A percent sign (%) after the route type indicates that RIP updates are always accepted for this destination.

DEST NET

Indicates the IP destination.

MASK

Indicates the IP destination subnet mask.

COST

Indicates the route cost:

<i>Table 30. MPROUTE IPv4 Route Type and COST Value mapping</i>	
Route Type	COST Value
SPF or SPIA	The OSPF cost of the route.
SPE1	The SPF cost to get to the AS boundary router or forwarding address that is used to reach the destination, plus the external cost.
SPE2	The external cost.
RIP	The RIP metric.
STAT or RSTA	• 0 when the route is direct. • 1 when the route is indirect.
DIR or SBNT	1
RNGE	The OSPF cost of the range.
DFLT	0

AGE

Indicates the time that has elapsed since the routing table entry was last refreshed.

NEXT HOP(S)

Indicates the IP address of the next router on the path toward the destination. A number in parentheses at the end of the column indicates the number of equal-cost routes to the destination. Use the `msg server_id rtable dest=ip-addr` command to obtain a list of the next hops.

Note: NETS DELETED and NETS INACTIVE are used only for internal debugging.

IPv4 Route expansion information: Use the `msg server_id rtable dest=ip-addr` command to obtain information about a particular route. When multiple equal-cost routes exist, use this command to obtain a list of the next hops.

Example: A sample output with explanation of entries follows:

```
msg mproutm0 rtable dest=10.0.2.0
Ready; T=0.01/0.01 09:31:57
EZZ7874I ROUTE EXPANSION
DESTINATION: 10.0.2.0
MASK: 255.255.255.0
ROUTE TYPE: SPF
DISTANCE: 2
AGE: 60982
NEXT HOP(S): 10.0.0.6 (M0TOM3)
              10.0.0.10 (M0TOM2)
```

DESTINATION

Indicates the IP destination.

MASK

Indicates the IP destination subnet mask.

ROUTE TYPE

Indicates how the route was derived:

DFLT

Indicates a route defined using the DEFAULT_ROUTE configuration statement in the MPRoute configuration file.

SBNT

Indicates that the network is subnetted; such an entry is a placeholder only.

DIR

Indicates a directly connected network, subnet, or host.

RIP

Indicates a route that was learned through the RIP protocol.

STAT

Indicates a nonreplaceable statically configured route.

SPF

Indicates that the route is an OSPF intra-area route.

SPIA

Indicates that the route is an OSPF interarea route.

SPE1

Indicates OSPF external routes (type 1).

SPE2

Indicates OSPF external routes (type 2).

RNGE

Indicates a route type that is an active OSPF area address range and is not used in forwarding packets.

RSTA

Indicates a static route that is defined as replaceable.

An asterisk (*) after the route type indicates that the route has a directly connected backup. A percent sign (%) after the route type indicates that RIP updates are always accepted for this destination.

DISTANCE

Indicates the route cost. For more information, see [Table 30 on page 293](#).

AGE

Indicates the time that has elapsed since the routing table entry was last refreshed.

NEXT HOP(S)

Indicates the IP address of the next router and the interface used to reach that router for each of the paths toward the destination.

IPv4 deleted routes information: Use the `msg server_id rtable deleted` command to obtain information about routes that were deleted from the MPRoute routing table and have not been replaced.

Example: A sample output follows. The entries displayed are described in [Configured IPv4 generic interfaces](#).

```
msg mprout1 rtable deleted
EZZ8137I IPV4 DELETED ROUTES
TYPE DEST NET MASK COST AGE NEXT HOP(S)
DEL 1.2.3.4 FFFFFFFF 16 12 NONE
1 NETS DELETED, 1 NETS INACTIVE
```

The following are examples of using the `SMSG IPV6OSPF` command.

All IPv6 OSPF information: The `msg server_id ipv6ospf all` command displays a comprehensive list of IPv6 OSPF information.

Example: A sample output with explanations of entries follows:

```
msg mproutm8 ipv6ospf all
Ready; T=0.01/0.01 09:38:19
EZZ7970I IPV6 OSPF INFORMATION
TRACE6: 0, DEBUG6: 0
STACK AFFINITY          TCPIP8
```

```

IPV6 OSPF PROTOCOL:      ENABLED
IPV6 OSPF ROUTER ID:     8.8.8.8 (*IPV6 OSPF)
DFLT IPV6 OSPF INST ID:  0
EXTERNAL COMPARISON:     TYPE 2
AS BOUNDARY CAPABILITY:  ENABLED
IMPORT EXTERNAL ROUTES:  RIP
ORIG. DEFAULT ROUTE:     ALWAYS
DEFAULT ROUTE COST:      (1,TYPE 2)
DEFAULT FORWARD. ADDR.:  ::
LEARN HIGHER COST DFLT:  NO
DEMAND CIRCUITS:         ENABLED
DR_MAX_ADJ_ATTEMPT:      0

EZZ7973I IPV6 OSPF AREAS
AREA ID      STUB DFLT-COST IMPORT-PREF DEMAND IFCS NETS RTRS ABRs
6.6.6.6      NO      N/A      N/A      ON      1      0      1      1
0.0.0.0      NO      N/A      N/A      ON      2      0      0      0

--AREA RANGES--
AREA ID      ADVERTISE  PREFIX
6.6.6.6      YES       E1:9::/64

EZZ7958I IPV6 OSPF INTERFACES
NAME          AREA          TYPE  STATE COST HELLO DEAD NBRS ADJS
M8TOGLAN9     0.0.0.0      BRDCST  8      1    10    40    0    0
M8TOGLAN6     6.6.6.6      BRDCST  32     1    10    40    3    2
VL/0          0.0.0.0      VLINK   1      1    30   180    1    0

EZZ7972I IPV6 OSPF VIRTUAL LINKS
ENDPOINT      TRANSIT AREA  STATE COST HELLO DEAD NBRS ADJS
7.7.7.7       6.6.6.6      1      1    30   180    1    0

EZZ8129I IPV6 OSPF NEIGHBORS
ROUTER ID     STATE LSRXL DBSUM LSREQ HSUP RTR-PRI IFC
7.7.7.7       8      0      0      0    OFF      1 M8TOGLAN6
10.10.10.10   128    1      0      0    OFF      1 M8TOGLAN6
9.9.9.9       128    1      0      0    OFF      1 M8TOGLAN6
7.7.7.7       1      0      0      0    OFF      0 *

```

TRACE6

Displays the level of tracing currently in use by MPRoute IPv6 routing protocols.

DEBUG6

Displays the level of debugging currently in use by MPRoute IPv6 routing protocols.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

IPV6 OSPF PROTOCOL

Displays whether IPv6 OSPF is enabled or disabled.

IPV6 OSPF ROUTER ID

Displays the IPv6 OSPF Router ID.

DFLT IPV6 OSPF INST ID

Displays the default value for the OSPF protocol instance identifier for IPV6_OSPF_INTERFACES.

EXTERNAL COMPARISON

Displays the external route type used by IPv6 OSPF when importing external information into the IPv6 OSPF domain and when comparing IPv6 OSPF external routes to IPv6 RIP routes.

AS BOUNDARY CAPABILITY

Indicates whether external routes will be imported into the IPv6 OSPF domain.

IMPORT EXTERNAL ROUTES

Indicates the types of external routes that will be imported into the IPv6 OSPF domain. Displayed only when AS Boundary Capability is enabled.

ORIG DEFAULT ROUTE

Indicates whether a default route will be originated into the IPv6 OSPF domain. Orig Default Route is displayed only when AS Boundary Capability is enabled.

DEFAULT ROUTE COST

Displays the cost and type of the default route (if originated). Default Route Cost is displayed only when AS Boundary Capability is enabled and Orig Default Route is Always.

DEFAULT FORWARD ADDR

Displays the forwarding address specified in the default route (if originated). Default Forwarding Address is displayed only when AS Boundary Capability is enabled and Orig Default Route is Always.

LEARN HIGHER COST DFLT

Indicates whether IPv6 OSPF will learn default routes from inbound packets when their cost is higher than the default route originated by this host. This parameter is only displayed when AS Boundary Capability is enabled and Orig Default Route is Always.

DEMAND CIRCUITS

Indicates whether demand circuit support is available for IPv6 OSPF interfaces.

DR_MAX_ADJ_ATTEMPT

Specifies a threshold value for the maximum number of adjacency attempts to a neighboring designated router. This value is used for reporting and controlling futile neighbor state loops. See [“Preventing futile neighbor state loops during adjacency formation”](#) on page 198.

The remainder of the `msg server_id ipv6ospf all` output is described in the following sections.

IPv6 OSPF area statistics and parameters: The `msg server_id ipv6ospf areasum` command displays the statistics and parameters for all IPv6 OSPF areas attached to the router.

Example: A sample output with an explanation of entries follows:

```
msg mproutm8 ipv6ospf areasum
Ready; T=0.01/0.01 09:38:19
EZZ7973I IPV6 OSPF AREAS
AREA ID      STUB DFLT-COST IMPORT-PREF DEMAND IFCS NETS RTRS ABRS
6.6.6.6      NO      N/A      N/A      ON      1      0      1      1
0.0.0.0      NO      N/A      N/A      ON      2      0      0      0

--AREA RANGES--
AREA ID      ADVERTISE PREFIX
6.6.6.6      YES      E1:9::/64
```

AREA ID

Indicates the ID of the area.

STUB

Indicates whether the area is a stub area.

DFLT-COST

Displays the cost of the default route configured for the stub area.

IMPORT-PREF

Indicates whether Inter-Area Prefix LSAs are to be imported into the stub area.

DEMAND

Indicates whether demand circuits are supported in this area. This is ON when every router in the area supports demand circuits, otherwise it is OFF.

IFCS

Indicates the number of router interfaces attached to the particular area. These interfaces are not necessarily functional.

NETS

Indicates the number of transit networks that have been found while doing the SPF tree calculation for this area.

RTRS

Indicates the number of routers that have been found when doing the SPF tree calculation for this area.

ABRS

Indicates the number of area border routers that have been found when doing the SPF tree calculation for this area.

AREA RANGES

Indicates that information about ranges configured for this area follows.

ADVERTISE

Indicates whether a given range within an area is to be advertised into other areas.

PREFIX

Displays the prefix and prefix length for a given range within an area.

IPv6 OSPF interface statistics and parameters: The `msg server_id ipv6ospf interface name=if-name id=if-id` command displays current run-time statistics and parameters related to IPv6 OSPF interfaces. (The keyword IF can be substituted for INTERFACE.) Either the NAME= parameter or the ID= parameter can be specified, but not both. If no NAME= or ID= parameter is given (see Example 1), a single line is printed summarizing each interface. If NAME= or ID= parameter is given (see Example 2), detailed statistics for that interface are displayed.

Examples: Sample outputs with explanations of entries follows:

```
1. msg mproutm8 ipv6ospf interface
Ready; T=0.01/0.01 09:21:14
EZZ7958I IPV6 OSPF INTERFACES
NAME          AREA          TYPE    STATE  COST  HELLO  DEAD  NBR5  ADJS
M8TOGLAN9     0.0.0.0       BRDCST   8      1    10    40    0     0
M8TOGLAN6     6.6.6.6       BRDCST  32      1    10    40    3     2
VL/0          0.0.0.0       VLINK    1      1    30   180    1     0
```

NAME

Displays the interface name.

AREA

Attached area ID.

TYPE

Can be one of the following:

BRDCST	Broadcast interface
VLINK	OSPF virtual link
VIPA	Virtual IP address link

STATE

Can be one of the following values. With the exception of Suspend, these values are described in RFC 1583 (OSPF Version 2).

1*	Suspend - The interface is suspended because an MSG command was issued or because it was unable to establish an adjacency with a neighboring designated router after it exceeded the futile neighbor state loop threshold DR_Max_Adj_Attempt). See “Preventing futile neighbor state loops during adjacency formation” on page 198.
1	Down
2	Backup
4	Looped back
8	Waiting
16	Point-to-point
32	DR other
64	Backup DR
128	Designated router

COST

Indicates the cost (or metric) associated with the interface.

HELLO

Indicates the number of seconds between Hello packets sent from the interface.

DEAD

Indicates the number of seconds after not having received an OSPF Hello packet, that a neighbor is declared to be down.

NBRS

Number of neighbors. This is the number of routers whose hellos have been received.

ADJS

Number of adjacencies. This is the number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.

```
2. smsg mproutm8 ipv6ospf interface name=m8toglan9
Ready: T=0.01/0.01 09:40:10
EZZ7959I IPV6 OSPF INTERFACE DETAILS
INTERFACE NAME:      M8TOGLAN9
INTERFACE ID:        2
INSTANCE ID:         0
INTERFACE ADDRESS:   FE80::209:5700:100:15
                     E1:9::9:8
INTERFACE PREFIX:    RADV E1:9::/64
ATTACHED AREA:      0.0.0.0
INTERFACE TYPE:      BRDCST
STATE:               128
DESIGNATED ROUTER:  8.8.8.8
BACKUP DR:           0.0.0.0

DR PRIORITY:         1    HELLO INTERVAL:    10    RXMT INTERVAL:     5
DEAD INTERVAL:       40    TX DELAY:         1    POLL INTERVAL:     60
DEMAND CIRCUIT:      OFF    HELLO SUPPRESS:    N/A    SUPPRESS REQ:      N/A
MTU:                 8192    COST:              1    DB_EX INTERVAL:    40

# NEIGHBORS:         0    # ADJACENCIES:     0    # FULL ADJS.:       0
# MCAST FLOODS:      0    # MCAST ACKS:       0    # MAX ADJ. RESETS:  0

NETWORK CAPABILITIES:
BROADCAST
DEMAND-CIRCUITS
MULTICAST
```

INTERFACE NAME

Displays the interface name.

INTERFACE ID

Number that uniquely identifies the interface among the collection of all OSPF interfaces on this TCP/IP stack.

INSTANCE ID

The IPv6 OSPF Instance ID for this interface.

INTERFACE ADDRESS

Indicates the IP addresses that have been learned from the TCP/IP stack for the interface.

INTERFACE PREFIX

Lists the interface's prefixes. RADV indicates the prefix was learned through IPv6 Router Discovery. STAT indicates it was statically defined to this interface using the PREFIX parameter of the IPV6_OSPF_INTERFACE statement. OSPF indicates it was learned using the OSPF protocol.

ATTACHED AREA

Attached area ID.

INTERFACE TYPE

Can be one of the following:

BRDCST	Broadcast interface
VLINK	OSPF virtual link
VIPA	Virtual IP address link

STATE

Can be one of the following values. With the exception of Suspend, these values are described in RFC 1583 (OSPF Version 2).

1*	Suspend - The interface is suspended because an SMSG command was issued or because it was unable to establish an adjacency with a neighboring designated router after it exceeded the futile neighbor state loop threshold DR_Max_Adj_Attempt). See “Preventing futile neighbor state loops during adjacency formation” on page 198.
1	Down
2	Backup
4	Looped back
8	Waiting
16	Point-to-point
32	DR other
64	Backup DR
128	Designated router

DESIGNATED ROUTER

Router ID of the designated router.

BACKUP DR

Router ID of the backup designated router.

DR PRIORITY

Displays the interface router priority used when selecting the designated router. A higher value indicates that this MPRoute is more likely to become the designated router. A value of 0 indicates that MPRoute will never become the designated router.

HELLO INTERVAL

Indicates the number of seconds between Hello packets sent from the interface.

RXMT INTERVAL

Displays the frequency (in seconds) of retransmitting link state update packets, link state request packets, and database description packets.

DEAD INTERVAL

Indicates the number of seconds after not having received an OSPF Hello packet, that a neighbor is declared to be down.

TX DELAY

Displays the transmission delay value (in seconds). As each link state advertisement is sent out through this interface, it will be aged by this value.

POLL INTERVAL

Displays the poll interval value.

DEMAND CIRCUIT

Displays the current demand circuit status.

HELLO SUPPRESS

Displays whether Hello Suppression is currently on or off.

SUPPRESS REQ

Displays whether Hello Suppression was requested for this interface.

MTU

Indicates the value of the Maximum Transmission Unit.

COST

Indicates the cost (or metric) associated with the interface.

DB_EX INTERVAL

Indicates the number of seconds to allow the database exchange to complete.

NEIGHBORS

Number of neighbors. This is the number of routers whose hellos have been received.

ADJACENCIES

Number of adjacencies. This is the number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.

FULL ADJS

Number of full adjacencies. This is the number of neighbors whose state is Full (and therefore with which the router has synchronized databases).

MAX ADJ. RESETS

Total number of times the maximum threshold value for attempting an adjacency with a neighboring designated router has been reset. The value N/A indicates that the field is not applicable for that interface, based on the interface type that is used to reach a neighbor. For more information, see the description of DR_MAX_ADJ_ATTEMPT in the [“Examples” on page 271](#) section (All OSPF configuration information).

MCAST FLOODS

Number of link state updates that flooded the interface (not counting retransmissions).

MCAST ACKS

Number of link state acknowledgments that flooded the interface (not counting retransmissions).

NETWORK CAPABILITIES

Displays the capabilities of the interface.

IPv6 OSPF virtual link statistics and parameters: The `msg server_id ipv6ospf vlink endpt=router-id` command displays current run-time statistics and parameters related to IPv6 OSPF virtual links. If no ENDPT= parameter is given (see Example 1), a single line is printed summarizing each virtual link. If ENDPT= parameter is given (see Example 2), detailed statistics for that virtual link will be displayed.

Examples: Sample outputs with explanations of entries follows:

```
1. msg mproutm8 ipv6ospf vlink
Ready: T=0.01/0.01 09:41:22
EZZ7972I IPV6 OSPF VIRTUAL LINKS
ENDPOINT      TRANSIT AREA  STATE  COST  HELLO  DEAD  NBRS  ADJS
7.7.7.7        6.6.6.6       1       1    30    180    1     0
```

ENDPOINT

Indicates the router ID of the virtual neighbor (other endpoint).

TRANSIT AREA

Indicates the non-backbone, non-stub area through which the virtual link is configured.

STATE

Can be one of the following:

1	Down
16	Point-to-point

For more information about these values, refer to RFC 1583 (OSPF Version 2).

COST

Indicates the cost (or metric) associated with the virtual link.

HELLO

Indicates the number of seconds between Hello packets sent from the virtual link.

DEAD

Indicates the number of seconds after not having received an OSPF Hello packet, that a neighbor is declared to be down.

NBRS

Number of neighbors. This is the number of routers whose hellos have been received.

ADJS

Number of adjacencies. This is the number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.

```
2. smsg mproutm8 ipv6ospf vlink endpt=7.7.7.7
Ready; T=0.01/0.01 09:41:22
EZZ7971I IPV6 VIRTUAL LINK DETAILS
VIRTUAL LINK ENDPOINT:      7.7.7.7
PHYSICAL INTERFACE NAME:    M8TOGLAN6
VL TRANSIT AREA:           6.6.6.6
STATE:                      16

HELLO INTERVAL:      30  DEAD INTERVAL:      180  DB_EX INTERVAL:      180
RXMT INTERVAL:       10  TX DELAY:           5    COST:              1
DEMAND CIRCUIT:      ON  HELLO SUPPRESS:    OFF  SUPPRESS REQ:        ON

# NEIGHBORS:          1  # ADJACENCIES:      1  # FULL ADJS.:         1
```

VIRTUAL LINK ENDPOINT

Indicates the router ID of the virtual neighbor (other endpoint).

PHYSICAL INTERFACE NAME

Indicates the name of the physical interface being used by the virtual link.

VL TRANSIT AREA

Indicates the non-backbone, non-stub area through which the virtual link is configured.

STATE

Can be one of the following:

1	Down
16	Point-to-point

For more information about these values, refer to RFC 1583 (OSPF Version 2).

HELLO INTERVAL

Indicates the number of seconds between Hello packets sent from the virtual link.

DEAD INTERVAL

Indicates the number of seconds after not having received an OSPF Hello packet, that a neighbor is declared to be down.

DB_EX INTERVAL

Indicates the number of seconds to allow the database exchange to complete.

RXMT INTERVAL

Displays the frequency (in seconds) of retransmitting link state update packets, link state request packets, and database description packets.

TX DELAY

Displays the transmission delay value (in seconds). As each link state advertisement is sent out through this interface, it will be aged by this value.

COST

Indicates the cost (or metric) associated with the virtual link.

DEMAND CIRCUIT

Displays the current demand circuit status.

HELLO SUPPRESS

Displays whether Hello Suppression is currently on or off.

SUPPRESS REQ

Displays whether Hello Suppression was requested for this interface.

NEIGHBORS

Number of neighbors. This is the number of routers whose hellos have been received.

ADJACENCIES

Number of adjacencies. This is the number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.

FULL ADJS

Number of full adjacencies. This is the number of neighbors whose state is Full (and therefore with which the router has synchronized databases).

IPv6 OSPF neighbor statistics and parameters: The `smmsg server_id ipv6ospf neighbor id=router-id ifname=if_name` command displays the statistics and parameters related to IPv6 OSPF neighbors. (The keyword NBR can be substituted for NEIGHBOR.)

- If no ID= parameter is given (see Example 1), a single line is printed summarizing each neighbor.
- If an ID= parameter is given (see Example 2), detailed statistics for that neighbor are displayed.
- If the neighbor specified by the ID= parameter has more than one neighbor relationship with MPRoute (for example if there are multiple IPv6 OSPF links connecting them), the IFNAME= parameter can be used to specify which link's adjacency to examine (for an adjacency over a virtual link, specify IFNAME=*).

Examples: A sample outputs with explanation of entries follows:

```
1. smmsg mproutm8 ipv6ospf neighbor
Ready; T=0.01/0.01 09:41:22
EZZ8129I IPV6 OSPF NEIGHBORS
ROUTER ID      STATE  LSRXL DBSUM LSREQ HSUP  RTR-PRI  IFC
7.7.7.7        8      0      0      0  OFF      1  M8TOGLAN6
10.10.10.10    128    1      0      0  OFF      1  M8TOGLAN6
9.9.9.9        128    1      0      0  OFF      1  M8TOGLAN6
7.7.7.7        1      0      0      0  OFF      0  *
```

ROUTER ID

Displays the neighbor's OSPF router ID.

STATE

Can be one of the following:

1	Down
2	Attempt
4	Init
8	2-Way
16	ExStart
32	Exchange
64	Loading
128	Full

For more information about these values, refer to RFC 1583 (OSPF Version 2).

LSRXL

Displays the size of the current link state retransmission list for this neighbor.

DBSUM

Displays the size of the database summary list waiting to be sent to the neighbor.

LSREQ

Displays the number of link state advertisements that are being requested from the neighbor.

HSUP

Displays whether hello suppression is active with the neighbor.

RTR-PRI

Displays the neighbor's router priority. Higher router priority indicates that it is more likely to become a designated router. A router priority of 0 indicates that the neighbor is not eligible to

become designated router. N/A indicates the neighbor is not on a multi-access link; therefore, no designated router is required.

IFC

Displays the name of the interface over which a relationship has been established with this neighbor. An asterisk (*) displayed in this column indicates that the neighbor relationship has been established over a virtual link.

```
2. msg mproutm8 ipv6ospf neighbor id=10.10.10.10
Ready; T=0.01/0.01 09:42:47
EZZ8130I IPV6 OSPF NEIGHBOR DETAILS
NEIGHBOR IP ADDRESS:    FE80::209:5700:100:1C
OSPF ROUTER ID:         10.10.10.10
NEIGHBOR STATE:         128
PHYSICAL INTERFACE:     M8TOGLAN6
DR CHOICE:               10.10.10.10
BACKUP CHOICE:           9.9.9.9
DR PRIORITY:             1
NBR OPTIONS:             V6,E,R (0X0013)

DB SUMM QLEN:           0  LS RXMT QLEN:           0  LS REQ QLEN:           0
LAST HELLO:             2  NO HELLO:           OFF
# LS RXMTS:              1  # DIRECT ACKS:         1  # DUP LS RCVD:         5
# OLD LS RCVD:           1  # DUP ACKS RCVD:        1  # ADJ. RESETS:         0
```

NEIGHBOR IP ADDRESS

Displays the link-local IP address of the neighbor's interface to the common link.

OSPF ROUTER ID

Displays the neighbor's OSPF router ID.

NEIGHBOR STATE

Can be one of the following:

1	Down
2	Attempt
4	Init
8	2-Way
16	ExStart
32	Exchange
64	Loading
128	Full

For more information about these values, refer to RFC 1583 (OSPF Version 2).

PHYSICAL INTERFACE

Displays the name of the interface over which a relationship has been established with this neighbor.

DR CHOICE, BACKUP CHOICE, DR PRIORITY

Indicate the values seen in the last hello received from the neighbor. N/A indicates that the neighbor is not on a multiaccess link; therefore, no designated router is required.

NBR OPTIONS

Indicates the optional OSPF capabilities supported by the neighbor. These capabilities are denoted by:

V6	The router can be used in IPv6 routing calculations.
E	Processes AS External LSAs. When this is not set, the area to which the common network belongs has been configured as a stub.

MC	RFC 1584 (Multicast Extensions to OSPF) is supported. This value is never set by MPRoute but can be received from other routers.
N	Describes the handling of Type-7 LSAs - Multicast OSPF. This value is never set by MPRoute but might be received from other routers.
R	Is an active router. Routes that transit the neighbor can be computed.
DC	RFC 1793 (Extending OSPF to Support Demand Circuits) is supported.

This field is valid only for those neighbors in state Exchange or greater.

DB SUMM QLEN

Indicates the number of advertisements waiting to be summarized in Database Description packets. It should be 0 except when the neighbor is in state Exchange.

LS RXMT QLEN

Indicates the number of advertisements that have been flooded to the neighbor, but not yet acknowledged.

LS REQ QLEN

Indicates the number of advertisements that are being requested from the neighbor in state Loading.

LAST HELLO

Indicates the number of seconds since a hello has been received from the neighbor.

NO HELLO

Indicates whether Hello Suppression is active with the neighbor.

LS RXMITS

Indicates the number of retransmissions that have occurred during flooding.

DIRECT ACKS

Indicates the number of acknowledgements sent in response to duplicate link state advertisements.

DUP LS RCVD

Indicates the number of duplicate retransmissions that have occurred during flooding.

OLD LS RCVD

Indicates the number of old advertisements received during flooding.

DUP ACKS RCVD

Indicates the number of duplicate acknowledgments received.

ADJ. RESETS

Indicates the number of times the neighbor has transitioned down to ExStart state.

IPv6 OSPF link state database statistics: The `msg server_id ipv6ospf dbsize` command displays the number of LSAs currently in the link state database, categorized by type.

Example: The following is a sample output:

```
msg mproutm8 ipv6ospf dbsize
Ready; T=0.01/0.01 09:43:21
EZZ8128I IPV6 OSPF LS DATABASE SIZE
# ROUTER-LSAS: 6
# NETWORK-LSAS: 2
# INTER-AREA PREFIX LSAS: 14
# INTER-AREA ROUTER LSAS: 1
# AS EXTERNAL-LSAS: 2
# LINK LSAS: 5
# INTRA-AREA PREFIX LSAS: 10
# UNKNOWN LSAS: 0
# INTRA-AREA ROUTES: 8
# INTER-AREA ROUTES: 0
# TYPE 1 EXTERNAL ROUTES: 0
# TYPE 2 EXTERNAL ROUTES: 2
```

IPv6 OSPF link state advertisement: The following command displays the contents of a single link state advertisement contained in the IPv6 OSPF database:

```
smsg server_id ipv6ospf lsa lstype=ls-type lsid=lsid orig=ad-router areaid=area-id
ifname=if_name
```

For a summary of all non-external advertisements in the IPv6 OSPF database, use the following command:

```
smsg server_id ipv6ospf database areaid=area_id
```

For a summary of all external advertisements in the IPv6 OSPF database, use the following command:

```
smsg server_id ipv6ospf external
```

Example: The following is a sample output of a Router LSA with explanations of entries:

```
EZZ7880I LSA DETAILS
      LS AGE:          767
      LS TYPE:         0X2001 (ROUTER LSA)
      LS ID:           0
      LS ORIGINATOR:   10.10.10.10
      LS SEQUENCE NO:  0X80000024
      LS CHECKSUM:     0X8DBC
      LS LENGTH:       56
      ROUTER TYPE:     (0X00)
      LS OPTIONS:      (0X000033) V6,E,R,DC
INTERFACES:
  TYPE  METRIC  INTERFACE ID    NBR INTERFACE ID    NBR ROUTER ID
    2      1         2      2    10.10.10.10
    2      1         1      1    10.10.10.10
```

LS AGE

The time, in seconds, since the LSA was originated. An asterisk (*) displayed beside the age value indicates that the originator is supporting demand circuits and has indicated that this LSA should not be aged.

LS TYPE

Classifies the advertisement and dictates its contents. LS Type values are hexadecimal values.

0x2001	Router LSA, has area scope.
0x2002	Network LSA, has area scope.
0x2003	Inter-Area Prefix LSA, has area scope.
0x2004	Inter-Area Router LSA, has area scope.
0x4005	AS External LSA, has global scope throughout the IPv6 OSPF autonomous system.
0x0008	Link LSA, has link scope.
0x2009	Intra-Area Prefix LSA, has area scope.

LS ID

Together with LS Type and LS Originator, uniquely identifies the LSA in the link state database.

LS ORIGINATOR

The Router ID of the router that originated the LSA.

LS SEQUENCE NO

Used to detect old or duplicate LSAs. Successive instances of an LSA are given successive LS sequence numbers.

LS CHECKSUM

The Fletcher checksum of the complete contents of the LSA, including the LSA header but excluding the LS age field.

LS LENGTH

The length in bytes of the LSA, including the 20-byte LSA header.

ROUTER TYPE

Indicates the level of function of the advertising router and can be one of the following:

ASBR	The router is an AS boundary router.
ABR	The router is an area border router.
V	The router is an endpoint of one of more fully adjacent virtual links having the described area as transit area.
W	The router is a wildcard multicast receiver (MPRoute will never set the W option on its own Router LSAs).

LS OPTIONS

Indicates the optional OSPF capabilities supported by the piece of the routing domain described by the advertisement, denoted by:

V6	The information in the LSA can be used in IPv6 routing calculations.
E	Processes AS External LSAs. When this is not set, the area to which the advertisement belongs has been configured as a stub.
MC	RFC 1584 (Multicast Extensions to OSPF) is supported. This value is never set by MPRoute but can be received from other routers.
N	Describes the handling of Type-7 LSAs - Multicast OSPF. This value is never set by MPRoute but can be received from other routers.
R	Routes can be computed which transit the advertising node.
DC	RFC 1793 (Extending OSPF to Support Demand Circuits) is supported.

INTERFACES

Subheader indicating that information about interfaces advertised on this Router LSA follows.

TYPE

The kind of interface being described:

1	Point-to-point connection to another router
2	Connection to a transit network
4	Virtual link

METRIC

The cost of using this router interface, for outbound traffic.

INTERFACE ID

The interface ID assigned to the interface being described.

NBR INTERFACE ID

The interface ID that the neighbor router (or, for Type 2 interfaces, the link's designated router) has been advertising in hello packets sent on the link.

NBR ROUTER ID

The Router ID of the neighbor router, or, for Type 2 interfaces, the link's designated router.

Example: The following is a sample output of a Network LSA with explanations of entries:

```
EZZ7880I LSA DETAILS
      LS AGE:          724
      LS TYPE:         0X2002 (NETWORK LSA)
      LS ID:           1
      LS ORIGINATOR:   10.10.10.10
      LS SEQUENCE NO:  0X80000028
```

```

LS CHECKSUM:      0X1711
LS LENGTH:        40
LS OPTIONS:       (0X000033) V6,E,R,DC
ATTACHED ROUTERS:
10.10.10.10      8.8.8.8      7.7.7.7      9.9.9.9

```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH, LS OPTIONS

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

ATTACHED ROUTERS

The Router IDs of each of the routers attached to the link. This includes the Designated Router and all routers that are fully adjacent to the Designated Router.

Example: The following is sample output of an Inter-Area Prefix LSA with explanations of entries:

```

EZZ7880I LSA DETAILS
LS AGE:      742
LS TYPE:     0X2003 (INTER-AREA PREFIX LSA)
LS ID:       5
LS ORIGINATOR: 7.7.7.7
LS SEQUENCE NO: 0X80000027
LS CHECKSUM: 0X78F5
LS LENGTH:   44
PREFIX:      E1:8::8:7/128
PREFIX-OPTIONS: (0X00)
METRIC:      0

```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

PREFIX

The prefix being described by the LSA.

PREFIX OPTIONS

The optional capabilities of the prefix including the following:

NU	The prefix should be excluded from IPv6 unicast calculations.
LA	The prefix is actually an IPv6 interface address of the advertising router.
MC	The prefix should be included in IPv6 multicast routing calculations.
P	On NSSA area prefixes, the prefix should be readvertised at the NSSA area border. MPRoute cannot be an NSSA area router.

METRIC

The cost of the route from the LSA originator to the prefix being described by the LSA.

Example: The following is sample output of an Inter-Area Router LSA with explanations of entries:

```

EZZ7880I LSA DETAILS
LS AGE:      *5
LS TYPE:     0X2004 (INTER-AREA ROUTER LSA)
LS ID:       1
LS ORIGINATOR: 8.8.8.8
LS SEQUENCE NO: 0X80000016
LS CHECKSUM: 0X199A
LS LENGTH:   32
LS OPTIONS:  (0X000033) V6,E,R,DC
ROUTER ID:   7.7.7.7
METRIC:      1

```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH, LS OPTIONS

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

ROUTER ID

The Router ID of the router being described by the LSA.

METRIC

The cost of the route from the LSA originator to the router being described by the LSA.

Example: The following is sample output of an AS External LSA with explanations of entries:

```
EZZ7880I LSA DETAILS
LS AGE:          1248
LS TYPE:         0x4005 (AS EXTERNAL LSA)
LS ID:           1
LS ORIGINATOR:   7.7.7.7
LS SEQUENCE NO:  0x8000001E
LS CHECKSUM:     0xA439
LS LENGTH:       28
METRIC:          1
METRIC TYPE:     2
PREFIX-OPTIONS:  (0x00)
PREFIX:          ::/0
```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

METRIC

The cost of the route from the LSA originator to the prefix being described by the LSA.

METRIC TYPE

Whether the specified metric is a Type 1 or Type 2 external metric.

PREFIX OPTIONS

The optional capabilities of the prefix including the following:

NU	The prefix should be excluded from IPv6 unicast calculations.
LA	The prefix is actually an IPv6 interface address of the advertising router.
MC	The prefix should be included in IPv6 multicast routing calculations.
P	On NSSA area prefixes, the prefix should be readvertised at the NSSA area border. MPRoute cannot be an NSSA area router.

PREFIX

The prefix being described by the LSA.

FORWARD ADDR

Optional field. If included, data traffic for the advertised destination should be forwarded to this address.

ROUTE TAG

Optional field. If included, communicates additional information between AS boundary routers.

REF TYPE, REF LS ID

Optional fields. If included, additional information concerning the advertised external route can be found in the LSA having LS type of REF TYPE, Link State ID of REF LS ID, and LS Originator the same as specified in this LSA.

Example: The following is a sample output of a Link LSA with explanations of entries:

```
EZZ7880I LSA DETAILS
LS AGE:          718
LS TYPE:         0x0008 (LINK LSA)
LS ID:           3
LS ORIGINATOR:   7.7.7.7
LS SEQUENCE NO:  0x80000018
LS CHECKSUM:     0x8659
LS LENGTH:       44
LS OPTIONS:      (0x000033) V6,E,R,DC
LINK LOCAL ADDR: FE80::209:5700:100:1D
ROUTER PRIORITY: 1
# PREFIXES:      0
```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH, LS OPTIONS

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

LINK LOCAL ADDR

The originating router's link-local address on the link.

ROUTER PRIORITY

The router priority of the interface attaching the originating router to the link. Used in electing Designated Router.

PREFIXES

The number of IPv6 address prefixes contained in the LSA.

PREFIX OPTIONS

The optional capabilities of the prefix:

NU	The prefix should be excluded from IPv6 unicast calculations.
LA	The prefix is actually an IPv6 interface address of the advertising router.
MC	The prefix should be included in IPv6 multicast routing calculations.
P	On NSSA area prefixes, the prefix should be readvertised at the NSSA area border. MPRoute cannot be an NSSA area router.

PREFIX

An IPv6 prefix to be associated with the link.

Example: The following is a sample output of an Intra-Area Prefix LSA with explanations of entries:

```
EZZ7880I LSA DETAILS
  LS AGE:          1117
  LS TYPE:         0X2009 (INTRA-AREA PREFIX LSA)
  LS ID:           0
  LS ORIGINATOR:   7.7.7.7
  LS SEQUENCE NO:  0X80000023
  LS CHECKSUM:     0XF735
  LS LENGTH:       52
  # PREFIXES:      1
  REF LS TYPE:     0X2001
  REF LS ID:       0
  REF ORIG:        7.7.7.7

METRIC  PREFIX-OPTIONS  PREFIX
0       (0X02) LA      E1:6::6:7/128
```

LS AGE, LS TYPE, LS ID, LS ORIGINATOR, LS SEQUENCE NO, LS CHECKSUM, LS LENGTH

See descriptions for these values in the Router LSA sample in [IPv6 OSPF link state advertisement](#).

PREFIXES

The number of IPv6 address prefixes contained in the LSA.

REF LS TYPE, REF LS ID, REF ORIG

Identifies the Router LSA or Network LSA with which the IPv6 address prefixes should be associated.

METRIC

The cost of the route from the LSA originator to each of prefixes being described.

PREFIX OPTIONS

The optional capabilities of each of the prefixes being described:

NU	The prefix should be excluded from IPv6 unicast calculations.
LA	The prefix is actually an IPv6 interface address of the advertising router.
MC	The prefix should be included in IPv6 multicast routing calculations.
P	On NSSA area prefixes, the prefix should be readvertised at the NSSA area border. MPRoute cannot be an NSSA area router.

PREFIX

The list of prefixes being described.

IPv6 OSPF external advertisements: The `msg server_id ipv6ospf external` command lists the AS external advertisements belonging to the IPv6 OSPF routing domain. One line is printed for each advertisement. Each advertisement is defined by the following three parameters:

- Its link state type (always 4005 for AS external advertisements)
- Its link state ID
- The advertising router (called the LS originator)

Example: A sample output with an explanation of entries follows:

```
smsg mproutm8 ipv6ospf external
Ready; T=0.01/0.01 09:55:08
EZZ8127I IPV6 OSPF AS EXTERNAL LSDB
      AS EXTERNAL LSAS (LS TYPE=4005)
LS ORIGINATOR  LS ID  SEQNO  AGE  PREFIX
7.7.7.7        1      0X80000023 1183  ::/0
7.7.7.7        2      0X80000023 1141  E1:5::/64
# ADVERTISEMENTS: 2  CHECKSUM TOTAL: 0X0000DFA0
```

LS ORIGINATOR

The Router ID of the router that originated the advertisement.

LS ID

Uniquely identifies multiple external LSAs originated by the same router.

SEQNO, AGE

It is possible for several instances of an advertisement to be present in the IPv6 OSPF routing domain at any one time. However, only the most recent instance is kept in the IPv6 OSPF link state database (and printed by this command). The LS sequence number (Seqno) and LS age (Age) fields are compared to see which instance is most recent. The LS age field is expressed in seconds. Its maximum value is 3600. An asterisk (*) displayed beside an age value indicates that the DONOTAGE bit is on.

PREFIX

The prefix being described by the LSA.

At the end of the display, the total number of AS external advertisements is printed, along with a checksum total over all of their contents. The checksum total is simply the 32-bit sum (carries discarded) of the individual advertisement LS checksum fields. This information can be used to quickly determine whether two IPv6 OSPF routers have synchronized databases.

IPv6 OSPF area link state database: The `msg server_id ipv6ospf database areaid=area-id` command displays the contents of a particular IPv6 OSPF area link state database. AS external advertisements are omitted from the display. A single line is printed for each advertisement. Each advertisement is defined by the following three parameters:

- Its link state type (called Type)
- The advertising router (called the LS originator)
- Its link state ID

Example: A sample output with an explanation of entries follows:

```
smsg mproutm7 ipv6ospf database areaid=0.0.0.0
Ready; T=0.01/0.01 10:03:39
EZZ8126I IPV6 OSPF AREA LS DATABASE
      ROUTER LSAS (LS TYPE=2001)
LS ORIGINATOR  LS ID  SEQNO  AGE  LINKS  RTR-TYPE
7.7.7.7        0      0X80000001 1495  1      ABR,ASBR
8.8.8.8        0      0X80000018 5* 1      ABR
# ADVERTISEMENTS: 2  CHECKSUM TOTAL: 0X0000CF8E

      INTER-AREA PREFIX LSAS (LS TYPE=2003)
LS ORIGINATOR  LS ID  SEQNO  AGE  PREFIX
7.7.7.7        4      0X80000024 1650  E1:6::6:7/128
7.7.7.7        6      0X80000023 1650  E1:6::6:A/128
7.7.7.7        7      0X80000023 1650  E1:7::7:A/128
7.7.7.7        8      0X80000023 1645  E1:6::6:9/128
7.7.7.7        9      0X80000023 1645  E1:7::7:9/128
7.7.7.7        17     0X80000001 1519  E1:6::6:8/128
```

```

8.8.8.8      4      0X80000017    5* E1:6::6:8/128
8.8.8.8      8      0X80000016    5* E1:6::6:7/128
8.8.8.8      9      0X80000016    5* E1:6::6:9/128
8.8.8.8     10      0X80000016    5* E1:7::7:9/128
8.8.8.8     11      0X80000001    29* E1:6::6:A/128
8.8.8.8     12      0X80000001    29* E1:7::7:A/128
# ADVERTISEMENTS: 12  CHECKSUM TOTAL: 0X00043E76

      INTER-AREA ROUTER LSAS (LS TYPE=2004)
LS ORIGINATOR  LS ID  SEQNO  AGE  DEST ROUTERID
8.8.8.8        1      0X80000016    5* 7.7.7.7
# ADVERTISEMENTS: 1  CHECKSUM TOTAL: 0X0000199A

      LINK LSAS (LS TYPE=0008)
LS ORIGINATOR  LS ID  SEQNO  AGE  INTERFACE
7.7.7.7        3      0X80000023 1691 M7TOGLAN8
# ADVERTISEMENTS: 1  CHECKSUM TOTAL: 0X000018C0

      INTRA-AREA PREFIX LSAS (LS TYPE=2009)
LS ORIGINATOR  LS ID  SEQNO  AGE  REF-LSTYPE  REF-LSID
7.7.7.7        3      0X80000023 1691 0X2001    0
8.8.8.8        2      0X80000016    5* 0X2001    0
# ADVERTISEMENTS: 2  CHECKSUM TOTAL: 0X00011D31

```

LS ORIGINATOR

The Router ID of the router that originated the advertisement.

LS ID

Uniquely identifies multiple LSAs of the same type originated by the same router.

SEQNO, AGE

It is possible for several instances of an advertisement to be present in the IPv6 OSPF routing domain at any one time. However, only the most recent instance is kept in the IPv6 OSPF link state database (and printed by this command). The LS sequence number (Seqno) and LS age (Age) fields are compared to see which instance is most recent. The LS age field is expressed in seconds. Its maximum value is 3600. An asterisk (*) displayed beside an age value indicates that the DONOTAGE bit is on.

LINKS

Number of links described by the LSA.

RTR TYPE

Indicates the level of function of the advertising router.

ASBR	The router is an AS boundary router.
ABR	The router is an area border router.
V	The router is an endpoint of one of more fully adjacent virtual links having the described area as transit area.
W	The router is a wildcard multicast receiver (MPRoute will never set the W option on its own Router LSAs).

ROUTERS

The number of routers attached to the link described by the LSA.

PREFIX

The prefix being described by the LSA.

INTERFACE

Associated interface.

REF LS-TYPE, REF-LS ID

Identifies the referenced Router LSA or Network LSA.

At the end of each type of LSA in the display, the total number of advertisements of that type in the area database is printed, along with a checksum total over all of their contents. The checksum total is simply the 32-bit sum (carries discarded) of the individual advertisement LS checksum fields. This information can be used to quickly determine whether two IPv6 OSPF routers have synchronized databases.

IPv6 OSPF router routes: The `msg server_id ipv6ospf routers` command displays all routes to other routers that have been calculated by IPv6 OSPF and are now present in the routing table.

Example: A sample output with explanations of entries follows:

```
smsg mproutm8 ipv6ospf routers
Ready; T=0.01/0.01 09:56:13
EZZ8125I IPV6 OSPF ROUTERS
DEST: 7.7.7.7
  NEXT HOP: FE80::209:5700:100:18
  DTYPE: BR RTYPE: SPF COST: 1 AREA: 6.6.6.6
DEST: 7.7.7.7
  NEXT HOP: FE80::209:5700:100:18
  DTYPE: ASBR RTYPE: SPF COST: 1 AREA: 6.6.6.6
DEST: 9.9.9.9
  NEXT HOP: FE80::209:5700:100:1A
  DTYPE: RTR RTYPE: SPF COST: 1 AREA: 6.6.6.6
DEST: 10.10.10.10
  NEXT HOP: FE80::209:5700:100:1C
  DTYPE: RTR RTYPE: SPF COST: 1 AREA: 6.6.6.6
DEST: 7.7.7.7
  NEXT HOP: E1::6:7
  DTYPE: BR RTYPE: SPF COST: 1 AREA: 0.0.0.0
```

DEST

Indicates the destination router's OSPF router ID.

NEXT HOP

Indicates the address of the next router on the path toward the destination host. A number in parentheses at the end of the address indicates the number of equal-cost routes to the destination.

DTYPE

Indicates the destination type:

ASBR

Indicates that the destination is an AS boundary router.

BR

Indicates that the destination is an area border router.

FADD

Indicates a forwarding address (for external routes).

RTR

Indicates that the destination is a router.

RTYPE

Indicates the route type and how the route was derived:

SPF

Indicates that the route is an intra-area route (comes from the Dijkstra calculation).

SPIA

Indicates that it is an inter-area route (comes from considering Inter-Area Router advertisements).

COST

Displays the cost to reach the router.

AREA

Displays the OSPF area to which the destination router belongs.

IPv6 OSPF routing protocol statistics: The `msg server_id ipv6ospf statistics` command displays statistics generated by the IPv6 OSPF routing protocol. (The keyword `STATS` can be substituted for `STATISTICS`.) The statistics indicate how well the implementation is performing, including its memory and network utilization.

Example: A sample output with explanations of entries follows:

```
smsg mproutm8 ipv6ospf statistics
Ready; T=0.01/0.01 09:56:42
EZZ8124I IPV6 OSPF STATISTICS
ATTACHED AREAS:                2  # DIJKSTRA RUNS:                4
```

OSPF PACKETS RCVD:	412	OSPF PACKETS RCVD W/ERRS:	1
TRANSIT NODES ALLOCATED:	24	TRANSIT NODES FREED:	16
LS ADV. ALLOCATED:	141	LS ADV. FREED:	97
QUEUE HEADERS ALLOC:	64	QUEUE HEADERS AVAIL:	64
INCREMENTAL SUMM. UPDATES:	6	INCREMENTAL VL UPDATES:	1
INCREMENTAL EXT. UPDATES:	0	PTRS TO INVALID LS ADV:	0
MULTICAST PKTS SENT:	276	UNICAST PKTS SENT:	19
LS ADV. AGED OUT:	3	LS ADV. FLUSHED:	20

ATTACHED AREAS

Indicates the number of areas to which the router has active interfaces.

DIJKSTRA RUNS

Indicates how many times the IPv6 OSPF routing table has been calculated from scratch.

OSPF PACKETS RCVD

Covers all types of IPv6 OSPF protocol packets.

OSPF PACKETS RCVD W/ERRS

Indicates the number of IPv6 OSPF packets that have been received that were determined to contain errors.

TRANSIT NODES

Allocated to store Router LSAs and Network LSAs.

LS ADV

Allocated to store Inter-Area Prefix, Inter-Area Router, AS External, Link, and Intra-Area prefix LSAs.

QUEUE HEADERS

Form lists of link state advertisements. These lists are used in the flooding and database exchange processes. If the number of queue headers allocated is not equal to the number available, database synchronization with a neighbor is in progress.

INCREMENTAL SUMM UPDATES, INCREMENTAL VL UPDATES

Indicates how many times new Inter-Area Prefix or Inter-Area Router LSAs have caused the routing table to be partially rebuilt.

INCREMENTAL EXT. UPDATES

Displays the number of changes to external destinations that are incrementally installed in the routing table.

PTRS TO INVALID LS ADV

Indicates the number of pointers on the LSA retransmit queue that point to invalid LS advertisements.

MULTICAST PKTS SENT

Covers IPv6 OSPF hello packets and packets sent during the flooding procedure.

UNICAST PKTS SENT

Covers IPv6 OSPF packet retransmissions and the Database Exchange procedure.

LS ADV. AGED OUT

Indicates the number of advertisements that have hit 60 minutes. Link state advertisements are aged out after 60 minutes. Usually they are refreshed before this time.

LS ADV. FLUSHED

Indicates the number of advertisements removed (and not replaced) from the link state database.

The following are examples of using the `SMSG IPV6RIP` command.

All IPv6 RIP information: The `smsg server_id ipv6rip all` command lists all IPv6 RIP-related information.

Example: A sample output with explanations of entries follows:

```
smsg mproutm6 ipv6rip all
Ready; T=0.01/0.01 10:13:55
EZZ8030I IPV6 RIP CONFIGURATION
TRACE6: 0, DEBUG6: 0
STACK AFFINITY: TCP6IPM6
IPV6 RIP: ENABLED
IPV6 RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
EZZ8027I IPV6 RIP INTERFACES
```

```

-----SEND----- --RCV--
NAME      MTU STATE IN  OUT PRF  HST STA DEF RADV PSN  PRF  HST
M6TOGLAN5 8192  UP   1   0 YES NO YES YES YES YES YES NO
M6TOGLAN4 8192  UP   1   0 YES NO YES YES YES YES YES NO

EZZ8031I IPV6 RIP ROUTE ACCEPTANCE
ACCEPT IPV6 RIP UPDATES ALWAYS FOR:
E1:5::5:5

EZZ8029I GLOBAL IPV6 RIP FILTERS
SEND ONLY: DEFAULT, DIRECT

IGNORE IPV6 RIP UPDATES FROM:
FE80::209:3200:200:12

FILTERS: NOSEND          E1:7::/64

```

TRACE6

Displays the level of tracing currently in use by MPRoute IPv6 routing protocols.

DEBUG6

Displays the level of debugging currently in use by MPRoute IPv6 routing protocols.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

IPV6 RIP DEFAULT ORIGINATION

Indicates the conditions under which IPv6 RIP supports default route generation and the advertised cost for the default route.

The remainder of the `msg server_id ipv6rip all` output is described in the following sections.

IPv6 RIP routes to be accepted: The `msg server_id ipv6rip accepted` command lists the routes to be unconditionally accepted, as configured with the `IPV6_ACCEPT_RIP_ROUTE` statement.

Example: A sample output with explanation of entries follows:

```

msg mproutm6 ipv6rip accepted
Ready; T=0.01/0.01 10:13:55
EZZ8031I IPV6 RIP ROUTE ACCEPTANCE
ACCEPT IPV6 RIP UPDATES ALWAYS FOR:
E1:5::5:5

```

ACCEPT IPV6 RIP UPDATES ALWAYS FOR

Indicates the prefixes and hosts for which updates are always accepted.

IPv6 RIP interface statistics and parameters: The `msg server_id ipv6rip interface` name=*if_name* command displays statistics and parameters related to IPv6 RIP interfaces. (The keyword IF can be substituted for INTERFACE.) If no NAME= parameter is given (`msg server_id ipv6rip interface`), a single line is printed summarizing each interface. (See example 1.) If a NAME= parameter is given, detailed statistics for that interface are displayed. (See example 2.)

Examples:

```

1. msg mproutm6 ipv6rip interface
Ready; T=0.01/0.01 10:13:55
EZZ8027I IPV6 RIP INTERFACES

NAME      MTU STATE IN  OUT PRF  HST STA DEF RADV PSN  PRF  HST
M6TOGLAN5 8192  UP   1   0 YES NO YES YES YES YES YES NO
M6TOGLAN4 8192  UP   1   0 YES NO YES YES YES YES YES NO

```

NAME

Indicates the name of the IPv6 RIP interface.

MTU

Indicates the value of the maximum transmission unit learned from the TCP/IP stack for the interface.

STATE

Indicates the status of the interface. Values are:

UP

The interface is up.

DOWN

The interface is known to TCP/IP but is down.

N/A

The interface is defined to MPRoute, but the TCP/IP stack has not informed MPRoute that the interface is installed. For detailed interface status information, use the NETSTAT DEVLINKS command.

IGNR

The interface is known to TCP/IP but is being ignored by MPRoute.

IN

Specifies the value of the metric to be added to IPv6 RIP routes received over this interface.

OUT

Specifies the value of the metric to be added to IPv6 RIP routes advertised over this interface.

SEND**PRF**

Indicates whether prefix routes are advertised in IPv6 RIP responses sent over this interface.

HST

Indicates whether host routes are advertised in IPv6 RIP responses sent over this interface.

STA

Indicates whether static routes are advertised in IPv6 RIP responses sent over this interface.

DEF

Indicates whether the default route, if available, is advertised in IPv6 RIP responses sent over this interface.

RADV

Indicates whether router advertisement routes are advertised in IPv6 RIP responses sent over this interface.

PSN

Indicates whether poisoned reverse routes are advertised in IPv6 RIP responses sent over this interface. A poisoned reverse route is one with an infinite metric (a metric of 16).

RCV**PRF**

Indicates whether prefix routes are accepted in IPv6 RIP responses received over this interface.

HST

Indicates whether host routes are accepted in IPv6 RIP responses received over this interface.

```
2. smsg mproutm6 ipv6rip interface name=m6toglan5
Ready; T=0.01/0.01 10:15:46
EZZ8028I IPV6 RIP INTERFACE DETAILS
INTERFACE NAME:      M6TOGLAN5
INTERFACE ADDRESS:   FE80::209:5700:100:13
                     E1:5::5:6
INTERFACE PREFIX:    RADV E1:5::/64
MTU:                 8192      STATE:                 UP
IN METRIC:           1        OUT METRIC:             0
SEND PREFIX ROUTES:  YES      SEND HOST ROUTES:  NO
SEND STATIC ROUTES:  YES      SEND DEFAULT ROUTES: YES
SEND RTR. ADV. ROUTES: YES    SEND POIS. REV. ROUTES: YES
RECEIVE PREFIX ROUTES: YES    RECEIVE HOST ROUTES: NO

SEND ONLY: ALL

FILTERS: NONE
```

INTERFACE NAME

Indicates the interface name.

INTERFACE ADDRESS

Indicates the IP addresses that have been learned from the TCP/IP stack for the interface.

INTERFACE PREFIX

Lists the interface prefixes. RADV indicates the prefix was learned through IPv6 Router Discovery. STAT indicates it was statically defined to this interface using the PREFIX parameter of the IPV6_RIP_INTERFACE statement.

MTU

Indicates the value of the maximum transmission unit learned from the TCP/IP stack for the interface.

STATE

Indicates the status of the interface. Values are:

UP

The interface is up.

DOWN

The interface is known to TCP/IP but is down.

N/A

The interface is defined to MPRoute, but the TCP/IP stack has not informed MPRoute that the interface is installed. For detailed interface status information, use the NETSTAT DEVLINKS command.

IGNORED

The interface is known to TCP/IP but is being ignored by MPRoute.

IN METRIC

Specifies the value of the metric to be added to IPv6 RIP routes received over this interface.

OUT METRIC

Specifies the value of the metric to be added to IPv6 RIP routes advertised over this interface.

SEND PREFIX ROUTES

Indicates whether prefix routes are advertised in IPv6 RIP responses sent over this interface.

SEND HOST ROUTES

Indicates whether host routes are advertised in IPv6 RIP responses sent over this interface.

SEND STATIC ROUTES

Indicates whether static routes are advertised in IPv6 RIP responses sent over this interface.

SEND DEFAULT ROUTES

Indicates whether the default route, if available, is advertised in IPv6 RIP responses sent over this interface.

SEND RTR. ADV. ROUTES

Indicates whether router advertisement routes are advertised in IPv6 RIP responses sent over this interface.

SEND POIS. REV. ROUTES

Indicates whether poisoned reverse routes are advertised in IPv6 RIP responses sent over this interface. A poisoned reverse route is one with an infinite metric (a metric of 16).

RECEIVE PREFIX ROUTES

Indicates whether prefix routes are accepted in IPv6 RIP responses received over this interface.

RECEIVE HOST ROUTES

Indicates whether host routes are accepted in IPv6 RIP responses received over this interface.

SEND ONLY

Indicates the route-type restrictions on IPv6 RIP sends for this interface.

FILTERS

Indicates the send and receive filters for this interface.

Global IPv6 RIP filters: The `msg_server_id ipv6rip filters` command displays the global IPv6 RIP filters.

Example: A sample output with explanations of entries follows:

```
smsg mproutm6 ipv6rip filters
Ready; T=0.01/0.01 10:15:46
EZZ8029I GLOBAL IPV6 RIP FILTERS
SEND ONLY: DEFAULT, DIRECT

IGNORE IPV6 RIP UPDATES FROM:
FE80::209:3200:200:12

FILTERS: NOSEND          E1:7::/64
```

SEND ONLY

Indicates the global route-type restrictions on IPv6 RIP sends that apply to all IPv6 RIP interfaces.

IGNORE IPV6 RIP UPDATES FROM

Indicates the IPv6 RIP routers from which advertisements will not be accepted.

FILTERS

Indicates the global send and receive filters that apply to all IPv6 RIP interfaces.

The following are examples of using the GENERIC6 command.

All IPv6 generic information: The `smsg server_id generic6 all` command lists all IPv6 generic information, which is information that is not specific to a routing protocol.

Example: A sample output with explanations of entries follows:

```
smsg mproutm6 generic6 all
Ready; T=0.01/0.01 10:19:33
EZZ8053I IPV6 GENERIC CONFIGURATION
TRACE6: 0, DEBUG6: 0
IPV6 TRACE DESTINATION: N/A
STACK AFFINITY: TCIPIM6

EZZ8060I IPV6 GENERIC INTERFACES
NAME          MTU STATE CONFIGURED
M6TOGLAN2     1500  UP    NO
M6TOGLAN7     8192  UP    YES
```

TRACE6

Displays the level of tracing currently in use by MPRoute IPv6 routing protocols.

DEBUG6

Displays the level of debugging currently in use by MPRoute IPv6 routing protocols.

IPV6 TRACE DESTINATION

Displays the file name of the IPv6 trace destination, or OMPCTRC if that destination is the MPRoute CTRACE.

Restriction: The trace destination will be displayed in upper case on the console, regardless of the case of the actual case-sensitive file name if the destination is an HFS file.

STACK AFFINITY

Displays the name of the stack on which MPRoute is running.

The remainder of the `smsg server_id generic6 all` output is described in the following sections.

IPv6 generic interface statistics and parameters: The `smsg server_id generic6 interface name=if-name` command displays statistics and parameters related to IPv6 generic interfaces. (The keyword IF can be substituted for INTERFACE.) If no NAME= parameter is given (`smsg server_id generic6 interface`), a single line is printed summarizing each interface. (See Example 1.) If a NAME= parameter is given, detailed statistics for that interface are displayed. (See Example 2.)

Examples:

1.

```
smsg mproutm6 generic6 interface
Ready; T=0.01/0.01 10:15:46
EZZ8060I IPV6 GENERIC INTERFACES
NAME          MTU STATE CONFIGURED
M6TOGLAN2     1500  UP    NO
M6TOGLAN7     8192  UP    YES
```

NAME

Indicates the name of the IPv6 generic interface.

MTU

Indicates the value of the maximum transmission unit learned from the TCP/IP stack for the interface.

STATE

Indicates the status of the interface. Values are:

UP

The interface is up.

DOWN

The interface is known to TCP/IP but is down.

N/A

The interface is defined to MPRoute, but the TCP/IP stack has not informed MPRoute that the interface is installed. For detailed interface status information, use the NETSTAT DEVLINKS command.

IGNR

The interface is known to TCP/IP but is being ignored by MPRoute.

CONFIGURED

Indicates whether or not the interface was configured to MPRoute.

```
2. smsg mproutm6 generic6 interface name=m6toglan7
Ready: T=0.01/0.01 10:24:14
EZZ8065I IPV6 GEN INTERFACE DETAILS
INTERFACE NAME:      M6TOGLAN7
INTERFACE ADDRESS:   FE80::209:5700:100:13
                     E1:7::7:6
INTERFACE PREFIX:    RADV E1:7::/64
MTU:                 8192
STATE:               UP
CONFIGURED:          YES
```

INTERFACE NAME

Indicates the interface name.

INTERFACE ADDRESS

Indicates the IP addresses that have been learned from the TCP/IP stack for the interface.

INTERFACE PREFIX

Lists the interface prefixes. RADV indicates the prefix was learned using IPv6 Router Discovery. STAT indicates it was statically defined to this interface using the PREFIX parameter of the IPV6_INTERFACE statement.

MTU

Indicates the value of the maximum transmission unit learned from the TCP/IP stack for the interface.

STATE

Indicates the status of the interface. Values are:

UP

The interface is up.

DOWN

The interface is known to TCP/IP but is down.

N/A

The interface is defined to MPRoute, but the TCP/IP stack has not informed MPRoute that the interface is installed. For detailed interface status information, use the NETSTAT DEVLINKS command.

IGNR

The interface is known to TCP/IP but is being ignored by MPRoute.

CONFIGURED

Indicates whether or not the interface was configured to MPRoute.

The following are examples of using the `SMMSG RT6TABLE` command.

MPRoute IPv6 routing table: The `smmsg server_id rt6table` command displays all of the routes in the MPRoute IPv6 routing table. A sample output with explanation of entries follows.



Attention: This command displays the contents of the working table that is used by MPRoute, not the TCP/IP routing table. The contents of the MPRoute routing table might contain different information than that in the TCP/IP routing table.

Example: A sample output with explanation of entries follows:

```
smmsg mproutm9 rt6table
Ready; T=0.01/0.01 10:29:43
EZZ7979I IPV6 ROUTING TABLE
DESTINATION: ::/0
  NEXT HOP: FE80::209:5700:100:18
  TYPE: SPE2          COST: 1          AGE: 3082
DESTINATION: E1:5::/64
  NEXT HOP: FE80::209:5700:100:18
  TYPE: SPE2          COST: 2          AGE: 3082
DESTINATION: E1:6::/64
  NEXT HOP: ::
  TYPE: RADV*         COST: 2          AGE: 64455
DESTINATION: E1:6::6:7/128
  NEXT HOP: FE80::209:5700:100:18
  TYPE: SPF           COST: 1          AGE: 64410
DESTINATION: E1:6::6:8/128
  NEXT HOP: FE80::209:5700:100:14
  TYPE: SPF           COST: 1          AGE: 3083
DESTINATION: E1:6::6:9/128
  NEXT HOP: ::
  TYPE: SPF*          COST: 0          AGE: 64410
DESTINATION: E1:6::6:A/128
  NEXT HOP: FE80::209:5700:100:1C (2)
  TYPE: SPF           COST: 1          AGE: 64406
DESTINATION: E1:7::/64
  NEXT HOP: ::
  TYPE: RADV*         COST: 2          AGE: 45
DESTINATION: E1:7::7:9/128
  NEXT HOP: ::
  TYPE: SPF*          COST: 0          AGE: 64410
DESTINATION: E1:7::7:A/128
  NEXT HOP: FE80::209:5700:100:1C (2)
  TYPE: SPF           COST: 1          AGE: 64406
DESTINATION: E1:8::8:7/128
  NEXT HOP: FE80::209:5700:100:18
  TYPE: SPIA          COST: 1          AGE: 3059
DESTINATION: E1:9::9:8/128
  NEXT HOP: FE80::209:5700:100:14
  TYPE: SPIA          COST: 1          AGE: 3058

DEFAULT GATEWAY IN USE.

TYPE COST      AGE      NEXT HOP
SPE2 1         3082     FE80::209:5700:100:18
                                0 NETS DELETED, 0 NETS INACTIVE
```

DESTINATION

Indicates the IP destination, along with its prefix length.

NEXT HOP

Indicates the IP address of the next router on the path toward the destination. A number in parentheses at the end of the column indicates the number of equal-cost routes to the destination. Use the `smmsg server_id rt6table dest=ip_addr` command to obtain a list of the next hops.

TYPE

Indicates how the route was derived:

DFLT

Indicates a route defined using the `IPV6_DEFAULT_ROUTE` configuration statement in the MPRoute configuration file.

DIR

Indicates a directly connected prefix or host.

RIP

Indicates a route that was learned through the IPv6 RIP protocol.

DEL

Indicates the route has been deleted.

Restriction: Deleted routes are shown only if RIP is active and only as long as RIP needs to advertise to neighboring routers that they have been deleted.

STAT

Indicates a nonreplaceable statically configured route.

SPF

Indicates that the route is an IPv6 OSPF intra-area route.

SPIA

Indicates that the route is an IPv6 OSPF interarea route.

SPE1

Indicates IPv6 OSPF external routes (type 1).

SPE2

Indicates IPv6 OSPF external routes (type 2).

RANGE

Indicates a route type that is an active IPv6 OSPF area address range and is not used in forwarding packets.

RSTA

Indicates a static route that is defined as replaceable.

RADV

Indicates a route that was learned by the TCP/IP stack through the IPv6 Router Discovery protocol.

An asterisk (*) after the route type indicates that the route has a directly connected backup. A percent sign (%) after the route type indicates that IPv6 RIP updates are always accepted for this destination.

COST

Indicates the route cost:

<i>Table 31. MPROUTE IPv6 Route Type and COST Value mapping</i>	
Route Type	COST Value
SPF or SPIA	The OSPF cost of the route.
SPE1	The SPF cost to get to the AS boundary router or forwarding address that is used to reach the destination, plus the external cost.
SPE2	The external cost.
RIP	The RIP metric.
STAT or RSTA	• 0 when the route is direct. • 1 when the route is indirect.
DIR or SBNT	1
RNGE	The OSPF cost of the range.
DFLT	0

Table 31. MPROUTE IPv6 Route Type and COST Value mapping (continued)

Route Type	COST Value
RADV	• 1 when the router advertisement indicated a preference of high. • 2 when the router advertisement indicated a preference of medium. • 3 when the router advertisement indicated a preference of low.

AGE

Indicates the time that has elapsed since the routing table entry was last refreshed.

Note: NETS DELETED and NETS INACTIVE are used only for internal debugging.

IPv6 Route expansion information: Use the `smsg server_id rt6table dest=ip_addr` command to obtain information about a particular IPv6 route. When multiple equal-cost routes exist, use this command to obtain a list of the next hops.

Example: A sample output with explanation of entries follows:

```
smsg mproutm9 rt6table dest
Ready; T=0.01/0.01 10:30:36
EZZ7980I IPV6 ROUTE EXPANSION
DESTINATION: E1:6::6:A/128
ROUTE TYPE:  SPF
COST:        1
AGE:         64459
NEXT HOP(S): FE80::209:5700:100:1C      (M9TOGLAN6)
               FE80::209:5700:100:1D      (M9TOGLAN7)
```

DESTINATION

Indicates the IP destination, along with its prefix length.

ROUTE TYPE

Indicates how the route was derived:

DFLT

Indicates a route defined using the `IPV6_DEFAULT_ROUTE` configuration statement in the MPRoute configuration file.

DIR

Indicates a directly connected prefix or host.

RIP

Indicates a route that was learned through the IPv6 RIP protocol.

STAT

Indicates a nonreplaceable statically configured route.

SPF

Indicates that the route is an IPv6 OSPF intra-area route.

SPIA

Indicates that the route is an IPv6 OSPF interarea route.

SPE1

Indicates IPv6 OSPF external routes (type 1).

SPE2

Indicates IPv6 OSPF external routes (type 2).

RANGE

Indicates a route type that is an active IPv6 OSPF area address range and is not used in forwarding packets.

RSTA

Indicates a static route that is defined as replaceable.

RADV

Indicates a route that was learned by the TCP/IP stack through the IPv6 Router Discovery protocol.

An asterisk (*) after the route type indicates that the route has a directly connected backup. A percent sign (%) after the route type indicates that IPv6 RIP updates are always accepted for this destination.

COST

Indicates the route cost. For more information, see [Table 31 on page 320](#).

AGE

Indicates the time that has elapsed since the routing table entry was last refreshed.

NEXT HOP(S)

Indicates the IP address of the next router and the interface used to reach that router for each of the paths toward the destination.

IPv6 deleted routes information: Use the `msg server_id rt6table deleted` command to obtain information about routes that were deleted from the MPRoute routing table and have not been replaced.

Example: Sample output follows. The entries displayed are described in [MPRoute IPv6 routing table](#).

```
msg mprout1 rt6table deleted
EZZ8137I IPV6 DELETED ROUTES
DESTINATION: 21:DB8:1::11:2:1/128
NEXT HOP: ::
TYPE: DEL COST: 1 AGE: 76484
DESTINATION: 21:DB8:1::12:2:1/128
NEXT HOP: ::
TYPE: DEL COST: 1 AGE: 76484
DESTINATION: 21:DB8:1::81:1:1/128
NEXT HOP: ::
TYPE: DEL COST: 1 AGE: 7656
DESTINATION: 21:DB8:1::87:1:1/128
NEXT HOP: ::
TYPE: DEL COST: 1 AGE: 7656
DESTINATION: 21:DB8:1::91:1:1/128
NEXT HOP: ::
TYPE: DEL COST: 1 AGE: 7656
NETS DELETED, 1 NETS INACTIVE
```

Example of stopping MPRoute: To stop MPRoute, issue `msg server_id shutdown`.

Example: For instance, if your server ID is MPROUT2, issue:

```
msg mprout2 shutdown
```

Example of rereading the configuration file: Use the `msg server_id reconfig` command to reread the MPRoute configuration file. This command ignores all statements in the configuration file except new OSPF_Interface, RIP_Interface, Interface, IPv6_RIP_Interface, IPv6_Interface, IPv6_OSPF_Interface, and IPv6_OSPF (ROUTERID parameter only) statements.

Example: For instance, if your server ID is MPROUT2, issue:

```
msg mprout2 reconfig
```

Rule: These new configuration statements must be reread from the configuration file through this command prior to any new interfaces referred to by new MPRoute configuration statements being configured to the TCP/IP stack.

Example of changing the cost of OSPF links: The cost of an OSPF interface can be dynamically changed using the `msg server_id ospf weight name=name cost=cost` command for an IPv4 OSPF interface or the `msg server_id ipv6ospf weight name=name cost=cost` command for an IPv6 OSPF interface. This new cost is flooded quickly throughout the OSPF routing domain, and modifies the routing immediately.

Example: For example, if your server ID is MPROUT2, you could change the cost of the IPv6 OSPF interface QDIO1 by issuing:

```
smsg mprout2 ipv6ospf weight name=qdio1 cost=20
```

The cost of the interface reverts to its configured value whenever MPRoute is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF interface in the configuration file.

Example of using the OPTIONS operand: For instance, if your server is MPROUT2, issue:

```
smsg mprout2 options
```

The OPTIONS operand lists all MPROUTE global configuration options information. The following contents show a sample output with an explanation of entries:

```
EZZ8173I GLOBAL OPTIONS
          IGNORE UNDEFINED INTERFACES:      YES
```

IGNORE UNDEFINED INTERFACES

Indicates whether MPROUTE will configure undefined interfaces with default values or neither configure nor advertise them.

Chapter 9. Configuring the NFS Server

The NFS server implements the Network File System (NFS), as well as PCNFSD user ID authentication function. To configure the NFS server, you must perform the following steps:

NFS Server Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the DTCPARMS file for the NFS server. 3. Establish NFS server machine authorizations. 4. Customize the VMNFS CONFIG file. 5. Configure NFS server file translation support. (Optional) 6. Verify NFS server operations. 7. Perform advanced NFS server configuration, if needed.

Dynamic Server Operation: The NFS server provides a VM Special Message (MSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information, see [“Dynamic Server Operation”](#) on page 342.

Step 1: Update PROFILE TCPIP

Include the NFS server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The NFS server is then started automatically when TCP/IP is initialized. The IBM default user ID for this server is **VMNFS**. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
VMNFS 0
```

The NFS server requires that ports TCP 2049 and UDP 2049 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
2049 UDP VMNFS ; NFS Server
2049 TCP VMNFS NOAUTOLOG ; NFS Server
```

Step 2: Update the DTCPARMS File

When the NFS server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the NFS server are:

```
:Nick.VMNFS
:ESM_Enable.
:ESM_Validate.
:ESM_Racroute.
:Anonymous.
:Paams.
:Timezone
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

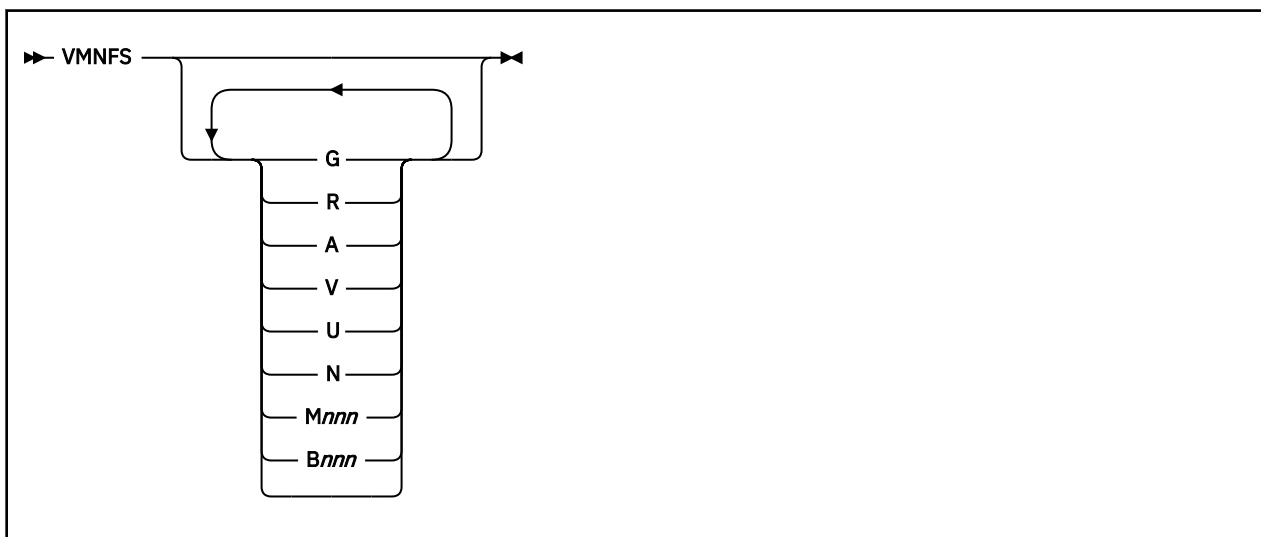
Note: Modify the DTCPARMS file for the NFS server in order to:

- Enable access to the server without requiring a VM user ID and password (anonymous access).
- Use an External Security Manager (ESM) for client authentication and minidisk access control.
- Access an SFS directory where trace information will be written using the SMSG TWRITE command.
- Change the CMS SET RECALL setting for the NFS server virtual machine to allow access to migrated SFS and BFS files.
- Set the correct timezone and allow the NFS server to correctly adjust for daylight savings time.

VMNFS Command Operands (:Parms. Parameters)

NFS services are initiated using the VMNFS command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with an NFS server definition. For more information about this command and its operands, see [“VMNFS Command Syntax” on page 326](#).

VMNFS Command Syntax



Operands

G

Causes a binary output file named VMNFS LOG A1 to be generated that contains a record of all RPC requests received and replies sent. Any existing log file is erased when the NFS server is started.

R

Indicates an external security manager (ESM) is to be used for access control. It is recommended that you specify :ESM_Enable .YES in the DTCPARMS file instead of providing this operand as a :Parms . tag startup parameter.

A

Causes Autolink access control to be used for mount requests.

V

Forces the NFS server to accept only Version 2 (RFC 1094) requests.

U

Forces the NFS server to accept only UDP connections. TCP connections are not permitted.

N

Indicates that ANONYMOUS mounts are permitted. It is recommended that you do not specify this operand as a :Parms . tag startup parameter, but instead specify :Anonymous .YES in the DTCPARMS file.

M *nnn*

Sets the trace mask to *nnn*, where *nnn* is a decimal number. The default trace mask is zero. For information about using trace masks to diagnose NFS server problems, consult the [z/VM: TCP/IP Diagnosis Guide](#).

B *nnn*

Sets the number of disk blocks, where *nnn* is a decimal number. The default number of disk blocks is 256.

Using an External Security Manager

The NFS server can use an external security manager (ESM) to authenticate NFS clients and to control access to minidisks by specifying `:ESM_Enable .Yes` in the DTCPARMS file. For more information, see Appendix A, “Using TCP/IP with an External Security Manager,” on page 685.

External Security Managers can protect SFS and BFS resources. However, that protection occurs in the file pool server machine, not in the NFS server. See [z/VM: CMS File Pool Planning, Administration, and Operation](#) for more information.

Step 3: Establish NFS Server Machine Authorizations

The system (CP) directory entry for the NFS server virtual machine must have `OPTION DIAG88` and `privilege class B` specified.

For NFS clients to access files or directories in the CMS Shared File System (SFS), the NFS server must have SFS file pool administrator authority. Each NFS server that will provide such access must be listed on the ADMIN statement in the SFS file pool server's DMSPARMS file. For details on SFS file pool configuration and administrator authority, see [z/VM: CMS File Pool Planning, Administration, and Operation](#).

For NFS clients to access files and directories in the Byte File System (BFS), the NFS server must have connect authority to the file pool, and the NFS server must be defined as a POSIX "superuser". To allow this capability, the following statements must be included in the CP directory entry for the NFS server virtual machine:

```
POSIXINFO UID 0 GID 0
POSIXOPT QUERYDB ALLOW
```

See [z/VM: OpenExtensions User's Guide](#) and [z/VM: CP Planning and Administration](#) for more information about configuring the NFS server in this manner.

Step 4: Customize the VMNFS CONFIG File

The NFS server configuration file, VMNFS CONFIG, contains configuration parameters for the NFS server. The statements used in this file specify:

- whether the PCNFSD function is available on the NFS server.
- whether NFS clients can request a list of all mounted file systems.
- the definition of the export list.
- whether NFS clients can mount other than what is defined in the export list.
- how many NFS clients using the TCP transport protocol can be concurrently handled by the NFS server.

See “NFS Configuration File Statements” on page 327 for detailed information about how to specify entries within this file.

NFS Configuration File Statements

NFS configuration file statements are processed when the NFS server is started. If you change one of these statements while the NFS server is in operation, the change will not be effective until the NFS server is restarted or an `SMSG M REFRESH CONFIG` is issued to the NFS server.

If the VMNFS CONFIG file cannot be found or cannot be opened, NFS server initialization continues, using the following default values:

- when the `lines=ext` or `trans=ext` keywords are used on a MOUNT, the default file extension values used are as defined by the TCPIP DATA file. See the [Chapter 3, “Defining the TCP/IP System Parameters,”](#) on page 13 for detailed information.
- the PCNFSD function is available on the VMNFS server
- NFS clients can obtain the list of all currently mounted file systems.
- There are no items in the export list.
- The maximum number of concurrent NFS clients using the TCP transport protocol is 50.

If the VMNFS CONFIG file cannot be read, the NFS server terminates with an error message.

Syntax Rules

The following syntax rules apply to statements specified in the NFS configuration file:

- Keywords (eg. EXPORT, etc.) are treated as if they were entered in uppercase.
- Variable operands are treated as if they were entered in uppercase *except* for EXPORT records, for which these operands are **case sensitive**.
- Configuration statements must begin and end on the same line.
- All comments must be preceded by a semicolon (;). A comment may follow a complete keyword and data specification on a record, or it may occupy a complete record.
- Blank records may be used to improve readability.

DUMPMOUNT Statement

The DUMPMOUNT statement determines whether the NFS server should make available to clients a list of all mounted file systems.



Operands

YES

Indicates the list of all mounted file systems is to be made available to clients. This is the default if a DUMPMOUNT statement is not specified in the NFS configuration file.

NO

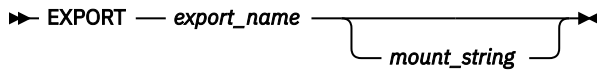
Indicates the list of all mounted file systems is *not* to be made available to clients.

Usage Notes

1. An NFS client requests the list of mounted file systems from the VMNFS server by sending a MNTPROC_DUMP request. If DUMPMOUNT YES is configured, the MNTPROC_DUMP reply contains the requested information.
2. DUMPMOUNT YES may make available to NFS clients the names of the SFS and BFS directories in your file pools. If you do not want to make this information available to any NFS client who can connect to your NFS server, specify DUMPMOUNT NO in the NFS server configuration file.
3. The information returned in a MNTPROC_DUMP reply includes resources actively in use by the VMNFS server. SFS and BFS directories are considered active if used within 15 minutes. A minidisk is considered active as long as the NFS server has that minidisk linked.

EXPORT Statement

The EXPORT statement defines an entry to be added to the NFS "export list", which consists of all EXPORT statements defined in the NFS configuration file. An NFS client can obtain the export list by using the OPENVM SHOWMOUNT command.



Purpose

Each EXPORT statement consists of a symbolic name that will be presented to NFS clients, optionally followed by the string that will be used when the NFS server receives a mount with a symbolic name.

Operands

export_name

The file system name that an NFS client can use to mount the *mount_string* (specified as the next operand). The *export_name* operand is **case sensitive** and is treated by the NFS client as a file system path name.

mount_string

An optional parameter. The *mount_string* identifies the file system to be mounted and any mount options that are to be substituted when the NFS server receives a mount request for *export_name*. The *mount_string* must be a syntactically valid mount command; that is, the object to be mounted (minidisk, SFS or BFS directory) followed by any options for the mount.

To assist in producing a valid *mount_string* the NFS server will recognize and process the following keywords within a *mount_string*:

%USERID

Causes the user ID that was specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure to be substituted in place of the %USERID keyword.

%FSROOT

Causes the FSROOT parameter of the POSIXINFO CP directory statement to be substituted in place of the %FSROOT keyword. The user ID specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure will be used to look up the VM CP directory entry.

%IWDIR

Causes the IWDIR parameter of the POSIXINFO CP directory statement to be substituted in place of the %IWDIR keyword. The user ID specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure will be used to look up the VM CP directory entry.

Usage Notes

1. If the EXPORTONLY YES statement is specified, then only the export list defined by the EXPORT records in the NFS configuration file can be mounted by NFS clients.
2. If an EXPORT statement does not specify the *mount_string* parameter, then the *export_name* must be a syntactically valid mount command with options, if any.
3. Due to operating system-specific conventions for displaying path names to files, it is recommended that an *export_name* consist of path name components that are eight or less characters. For example, use /PC/Your/SFS/In/FPC00L as opposed to PC/Your_SFS_In_FPC00L.
4. For examples of EXPORT statements, consult the sample NFS configuration file, VMNFS SCONFIG.
5. To export BFS directories whose names include spaces, place the *mount_string* within double quotation marks(""). An export name can also include spaces if it is surrounded by double quotation marks. Keep in mind that some NFS clients require special syntax to mount an alias with quotation

marks and/or spaces. For example, one client requires the following syntax to mount the directory named “my directory” to mydir:

```
mount gd3vm0:"my\directory\" mydir
```

- The NFS server ignores any export entries that have aliases which have already been used by another export entry. Initial slashes (/) are interpreted as being nonexistent, so the aliases “/alias” and “alias” are considered equivalent.

EXPORTONLY Statement

The EXPORTONLY statement restricts the file systems that can be mounted by NFS clients to the list of EXPORT records defined in the NFS configuration file.



Operands

NO

Indicates there are no restrictions on the file systems that can be mounted by an NFS client. In this case the export list augments what can be mounted. This is the default if the EXPORTONLY statement is not specified in the NFS configuration file.

YES

Indicates that NFS clients can mount only those file systems which are identified by EXPORT statements within the NFS configuration file.

Usage Notes

- If EXPORTONLY YES is specified but no EXPORT records are defined, **and**:
 - the NFS server is being initialized with this configuration, the server terminates with an error message.
 - this configuration is put into place using an SMSG M REFRESH CONFIG command, then the NFS server will not allow any new mounts. However, the use of previously mounted file systems remains unaffected. A warning message is displayed on the server console indicating that the NFS server is in this state. To resume handling new mount requests, a new NFS configuration file should be reloaded in which either EXPORTONLY NO or an export list is defined.

MAXTCPUSERS Statement

The MAXTCPUSERS statement specifies the maximum number of NFS clients using the TCP transport protocol that can be concurrently supported by the NFS server.



Operands

maxtcpusers_value

Defines the value to be used. If a MAXTCPUSERS statement is not specified in the NFS configuration file, the NFS server defaults to supporting 50 concurrent NFS clients that use the TCP transport protocol.

Usage Notes

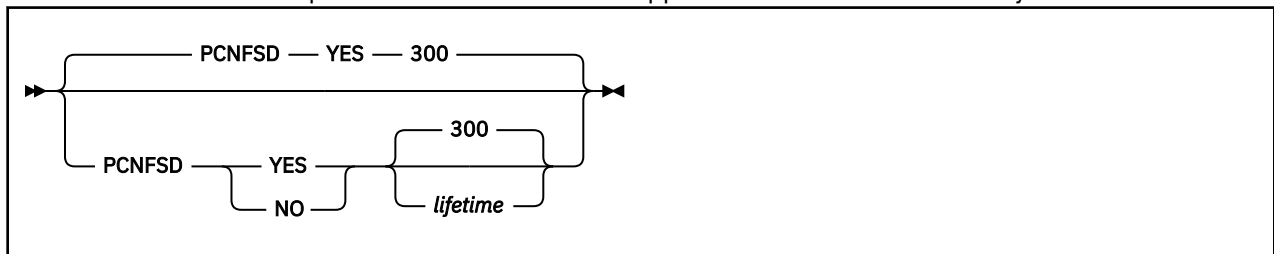
1. The SMSG M REFRESH CONFIG command cannot be used to put a new *maxtcpusers_value* into effect while the NFS server is in operation. To activate the new value, the NFS server must be stopped and restarted.
2. For installations whose NFS clients are configured to use the TCP transport protocol, the default *maxtcpusers_value* may need to be increased. Otherwise, some of these users may be unable to access the NFS server. They may experience errors indicating connections were refused or not allowed. Since it may be difficult to determine how many users rely on use of the TCP transport protocol, it may be desirable to initially configure an arbitrarily large *maxtcpusers_value* and then revise this value over time.

To assist in tuning the *maxtcpusers_value*, an SMSG M QUERY CONFIG command can be used to query the *maxtcpusers_value* currently in effect and the current number of NFS clients using TCP transport protocol. Note that values returned in the response reflect the current instance of the NFS server since it was started or restarted. For more information about this SMSG command, see the chapter, "Using the Network File System Commands", in the *z/VM: TCP/IP User's Guide*.

3. The MAXTCPUSERS value used by the NFS server may be lower than the *maxtcpusers_value* specified. Message DTCNFS1553I is displayed on the VM NFS server console during its initialization to indicate this has occurred. There are two reasons why this can occur:
 - The NFS server requires a small number of socket connections for its use. Therefore, if *maxtcpusers_value* prevents the NFS server from obtaining these connections, MAXTCPUSERS will be set to a lower value.
 - The number of socket interface control blocks (SKCBs) specified with the SKCBPOOLSIZE statement in the TCPIP server configuration file may need to be increased in order for the desired *maxtcpusers_value* to be used.

PCNFSD Statement

The PCNFSD statement specifies whether PCNFSD support is to be made available by the NFS server.



Operands

YES

Indicates that PCNFSD support is made available by the NFS server. The default is PCNFSD YES 300, which causes PCNFSD information to be discarded after 5 minutes (300 seconds). Specifying PCNFSD YES removes the need for a user ID and password to be supplied on mountpw and mount requests that are submitted by clients that have PCNFSD support.

NO

Indicates that PCNFSD support is to not be made available by the NFS server.

lifetime

The number of seconds the NFS server should keep PCNFSD information active after a PCNFSD request is received from an NFS client. The default is 300 seconds (5 minutes). After the specified *lifetime*, the PCNFSD information is discarded.

VMFILETYPE Statement

The VMFILETYPE statement is supported within the NFS server configuration file only to maintain compatibility with prior levels of TCP/IP for z/VM. File extension support should be configured through use of an equivalent VMFILETYPE statement within the TCPIP DATA file.

Purpose

For detailed information about VMFILETYPE operands, see [Chapter 3, “Defining the TCP/IP System Parameters,”](#) on page 13.

Note: The syntax for a VMFILETYPE statement within the TCPIP DATA file differs slightly compared to that for the NFS server configuration file.

Step 5: Configure NFS Server File Translation Support (Optional)

By default, the NFS server does not manipulate the file data it processed in response to client requests — that is, no EBCDIC-ASCII data translation is performed and line feed characters are not inserted at CMS record boundaries when files are processed. However, the NFS server can be configured to perform these actions for specific types of files, based on a file *extension* (or with respect to CMS files, the file *type*) of a file that is referenced. This can simplify NFS operations for various users and clients, and may even be necessary to accommodate certain NFS clients.

To configure file translation support for the NFS server, customize the TCPIP DATA file to include the appropriate VMFILETYPE and VMFILETYPEDEFAULT statements. The NFS server relies upon these statements to control the manner in which file translation and line feed processing are performed for specific file extensions, as well as those that are "unknown" or not recognized. For detailed information about how to specify these statements, see [Chapter 3, “Defining the TCP/IP System Parameters,”](#) on page 13.

Note:

1. The VMFILETYPE statement determines whether EBCDIC-ASCII translation occurs for a specific file and whether line feed characters are inserted at CMS record boundaries, based on the extension of that file.
2. File extensions that are not dealt with through a specific VMFILETYPE statement are considered as "unknown" (that is, are not recognized). The translation performed for such files (if any) is controlled by the VMFILETYPEDEFAULT statement. If the VMFILETYPEDEFAULT statement is not used, no translation is performed and no line feed characters are inserted at CMS record boundaries.
3. Case (upper or lower) is not significant when the NFS server compares the *filetype* (*extension*) supplied in an NFS request with those on *filetype* VMFILETYPE statements. For example, BIN is considered to be equivalent to Bin.

For additional information about how NFS clients can make use of file translation, see the [z/VM: TCP/IP User's Guide](#).

Step 6: Verify NFS Server Operations

After the NFS server has successfully initialized, use the steps that follow to verify that the server is configured and running correctly:

1. From another system, issue a PING command against the z/VM host to verify that network connectivity has been established. For example:

```
ping vmsys1.endicott.ibm.com
```


2. Issue an `RPCINFO` command to verify that the Portmapper server is up and running. The `RPCINFO` command can be issued from any platform that supports this command. For a z/VM host, the `RPCINFO` command format is as follows:

```
rpcinfo -p server_name
```

For example:

```
rpcinfo -p vmsys1.endicott.ibm.com
```

The Portmapper service should respond with a list of programs, versions, protocols, and port numbers. This response should be similar to the following:

```
rpcinfo -p vmsys1
  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, start the Portmapper server or have this done by the appropriate system administration personnel.

3. If your users mount using 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 operating correctly on your VM system by entering the following VM `RPCINFO` commands:

```
rpcinfo -u host_name mount
rpcinfo -u host_name pcnfsd
rpcinfo -u host_name portmapper
rpcinfo -u host_name nfs
```

If these 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 vmsys1 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 vmsys1 portmapper
program 100000 version 2 ready and waiting
Ready; T=0.04/0.08 16:46:40

* See if nfs is running
rpcinfo -u vmsys1 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 vmsys1 pcnfsd
program 150001 version 1 ready and waiting
program 150001 version 2 ready and waiting
Ready; T=0.04/0.08 16:47:53
```

5. Test the NFS server by issuing a mount request from a remote client. If the NFS server terminates with a return code of 101 or 102, a request to allocate a control block (101), or a disk block (102) buffer failed.
6. Verify the correct VMNFS CONFIG file is in effect and that this file has been customized as intended. To do this, issue the following command:

```
smsg server_id m query config
```

to the NFS server from an appropriately authorized user ID, such as TCPMAINT. The response for this command can be reviewed to verify that the EXPORTONLY statement has been correctly specified to allow or disallow users from performing mounts. A sample SMSG QUERY CONFIG command response follows:

```
smsg vmnfs m query config
Ready; T=0.01/0.01 16:39:54
16:39:54 * MSG FROM VMNFS : M Exportonly no, Anonymous yes, Dumpmount yes, Security CP
16:39:54 * MSG FROM VMNFS : M PCNFSD yes, time out after 300 seconds
16:39:54 * MSG FROM VMNFS : M UDP V3 buffer sizes: 8192 read, 8192 write, 8192 preferred.
16:39:54 * MSG FROM VMNFS : M TCP V3 buffer sizes: 65536 read, 65536 write, 32768
preferred.
16:39:54 * MSG FROM VMNFS : M Maximum TCP clients: 50.
16:39:54 * MSG FROM VMNFS : M No NFS clients are currently using TCP.
```

Step 7: Advanced Configuration Considerations

Before you complete the NFS server configuration process, you may want to review the information in the following sections to determine if additional NFS configuration is appropriate or necessary for your installation.

Prior to customizing the server exits described in this section, ensure that you have reviewed the exit limitations and customization recommendations presented in [“Customizing Server-specific Exits”](#) on page 49.

NFS Server Exits

PI

Several server exits are supported for use with the NFS server that allow for greater control over client mount requests, as well as client and administrative interaction with the server. These exits are described in more detail in the sections that follow.

CMS Command Exit

The CMS command exit (VMNFSCMS EXEC) provides authorization and execution control over CMS commands issued to the NFS server through the SMSG interface.

For example, CMS commands similar to the following might be issued by the TCPMAINT user ID to obtain the NFS server console spool file while the server is operating:

```
cp smsg vmnfs m cms cp sp con to tcpmaint
cp smsg vmnfs m cms cp close con
cp smsg vmnfs m cms identify
```

Keep in mind the following when the CMS command exit is customized:

Note:

1. Any security guidelines or authorization controls established for your site should be incorporated within the exit before it is activated.
2. Consideration should also be given to traditional security issues when this exit is modified and enabled. For example, the NFS server requires the use of privileged commands that are denied to general users, and may have access to files that contain passwords (such as the VMNFS HISTORY file). Measures should be taken to ensure areas such as these are properly addressed.

3. Care should be taken so that commands or programs that may adversely affect the NFS server are not permitted. For example, a command that overlays the VMNFS MODULE or that causes CMS storage management changes should be restricted from use. Likewise, a program that uses TCP/IP services — even if it does not create a storage conflict — is likely to cause operational problems.
4. The NFS server utilizes the first command exit found in its CMS search order. If a CMS command exit is not available to the NFS server, it rejects all SMSG CMS commands.

CMS Command Exit Input

For each SMSG CMS command that it receives, the NFS server stores values pertinent to that command in several GLOBALV variables (maintained in the GLOBALV group VMNFS) which can be referenced within the CMS command exit. These variables, and a description of the value each may have, are listed here:

Variable

Description

ORIGIN_NODE

The name of the original sending host, if the SMSG is delivered through RSCS. An asterisk (*) is supplied if the command is not from RSCS.

ORIGIN_USERID

The virtual machine that originated the SMSG command, if the SMSG is delivered through RSCS. An asterisk (*) is supplied if the command is not from RSCS.

REPLY_TAG

The *replytag* supplied to the NFS server for this SMSG command.

VERB

The CMS operand of the SMSG command. For this exit, 'CMS' is a constant that does not change.

ARGS

The CMS command arguments supplied with the SMSG CMS command.

SMSG

The complete, unmodified SMSG command text.

The following are passed to the CMS command exit as command line arguments:

Argument	Value	Format
1	CMS command arguments as supplied with the SMSG CMS command.	Character

Return Codes

The NFS server does make use of the return code established by this exit. However, a nonzero return code will cause an informational message and a list of GLOBALV variables and values to be displayed at the NFS server console.

CMS Command Exit Sample

A sample CMS command exit is provided as VMNFS CMS SAMPEXEC on the TCPMAINT 591 minidisk. Your customized exit should be maintained on the TCPMAINT 198 minidisk, with a file type of EXEC. For more information about the supplied CMS command exit, review the content of the VMNFS CMS SAMPEXEC file.

Mount Monitor Exit

The mount monitor exit (VMNFS MON EXEC) provides the ability to monitor or control all client mount requests. The monitor exit allows these requests to be accepted or rejected based on information about the requesting user or client, such as the client host IP address. In addition, the monitor exit may modify certain mount string values supplied by a client before mount processing is completed.

Keep in mind the following when the mount monitor exit is customized:

Note:

1. The NFS server utilizes the first mount monitor exit found in its CMS search order. If a mount monitor exit is not available to the NFS server, it processes all mount requests based only on client-supplied mount parameters.

Mount Monitor Exit Input

For each mount request that it receives, the NFS server stores values pertinent to that request in several GLOBALV variables (maintained in the GLOBALV group VMNFS) which can be referenced within the mount monitor exit. These variables, and a description of the value each may have, are listed here:

Variable Description

ACCOUNT

The account identification string provided with the mount request via the account= operand.

CLIENT

The IP address (in dotted-decimal format) of the NFS client host system that issued the mount request. For example: 9.130.48.134

FILESYSTEM

The minidisk, SFS, or BFS resource to be mounted through this mount request. For example, ELWOOD.191 might be supplied for a minidisk mount request, whereas FPCOOL:ELWOOD. might be supplied for an SFS mount, and /. /VMBFS:FPCOOL:ROOT/home/elwood/ might be supplied for a BFS request.

LINKPASSWORD

The minidisk link password supplied with a minidisk mount request, via the mdiskpw= operand. For SFS and BFS mount requests, the LINKPASSWORD value is not applicable and is not defined.

RECORD

A decimal value that corresponds to the presence of trans=, lines=, and record= operands in the mount request. Possible RECORD variable values follow:

RECORD Value

Associated Mount Operands

136

trans=ext and lines=ext

132

trans=ext and lines=n1

130

trans=ext and lines=CMS

129

trans=ext and lines=none

72

trans=yes and lines=ext

68

trans=yes and lines=n1

66

either (trans=yes and lines=CMS) or (record=text)

65

trans=yes and lines=none

40

trans=no and lines=ext

36

trans=no and lines=n1

34

either (trans=no and lines=CMS) or (record=binary)

33

trans=no and lines=none

TYPE

A decimal value that corresponds to the type of mount request. Possible TYPE variable values follow:

TYPE Value**Mount Type****17**

Read-write mount request

18

Read-only mount request

USERID

The logon user ID specified with the mount request via the `userid=` operand, or through a PCNFSD request.

BYUSERID

The LOGONBY user ID specified with the mount request via the `by=` operand.

Values for the ACCOUNT, FILESYSTEM and LINKPASSWORD variables can be modified within the mount monitor exit. Changes to values maintained by these GLOBALV variables are then used by the NFS server when the monitor exit returns processing control.

Note: If the FILESYSTEM value is altered, the *type* of file system that was specified by the requesting client must be maintained. For example, if FILESYSTEM originally represents a minidisk, any changed value must continue to represent a minidisk — an SFS or BFS directory cannot be substituted in such a case.

No command line arguments are passed to the mount monitor exit.

Return Codes

The NFS server recognizes the following exit return codes:

Return Code	Meaning
0	Continue and process the mount request
<i>nonzero</i>	Deny the mount request and return "not authorized" status to the client

Mount Monitor Exit Sample

A sample mount monitor exit is provided as VMNFSSMON SAMPEXEC on the TCPMAINT 591 minidisk. Your customized exit should be maintained on the TCPMAINT 198 minidisk, with a file type of EXEC. For more information about the supplied mount monitor exit, review the content of the VMNFSSMON SAMPEXEC file.

SMSG Authorization Exit

The SMSG authorization exit (VMNFSSMSG EXEC) provides the ability to control the acceptance and processing of SMSG commands issued to the NFS server, based on the type and purpose of an SMSG command, as well as the originator of the command.

Keep in mind the following when the SMSG authorization exit is customized:

Note:

1. Any security guidelines or authorization controls established for your site should be incorporated within the exit before it is activated.
2. The NFS server utilizes the first SMSG authorization exit found in its CMS search order. If an SMSG authorization exit is not available to the NFS server, it accepts and processes all SMSG commands (with SMSG CMS command processing controlled by the CMS command exit, or lack thereof).

SMSG Authorization Exit Input

For each SMSG command that it receives, the NFS server invokes the SMSG authorization exit, with the following passed as command line arguments:

Argument	Value	Format
1	User ID that originated the SMSG command	Character
2	Node ID from which the SMSG command was issued	Character
3	Text of the supplied SMSG	Character

Return Codes

The NFS server recognizes the following exit return codes:

Return Code	Meaning
0	Accept and process the supplied SMSG command.
<i>nonzero</i>	Do not process the supplied command — the originating user ID is not authorized.

SMSG Authorization Exit Sample

A sample SMSG authorization exit is provided as VMNFSSMG SAMPEXEC on the TCPMAINT 591 minidisk. Your customized exit should be maintained on the TCPMAINT 198 minidisk, with a file type of EXEC. For more information about the supplied SMSG authorization exit, review the content of the VMNFSSMG SAMPEXEC file.

PI end

Managing Translation Tables

Many different translation tables can be made available to clients, in order to facilitate the proper ASCII/EBCDIC translation of data. The specific translation table to be used in processing data for a given client is specified as part of its MOUNT request, through use of the `xlate=tablename` operand. If the corresponding *tablename* TCPXLBIN file is present in the search order for the NFS server, that translation table is then used; if such a file is not available, the mount attempt fails. For more information, see [Chapter 19, "Using Translation Tables," on page 667](#).

A maximum of 255 translation tables can be accommodated by the NFS server. If this maximum is exceeded, a notification message is displayed at the NFS server console and an I/O error is returned to any client that requests the use of a translation table which is not already active.

The mapping of translation tables currently in use by the NFS server can be "refreshed" through the following actions:

1. Stop the NFS server
2. Erase the VMNFS TRANSLAT file that resides on the server 191 minidisk.
3. Restart the NFS server.

Note:

1. Do not attempt to directly modify the VMNFS TRANSLAT file, as this can corrupt its content.
2. When the translation table mapping is refreshed in this manner, the VMNFS HISTORY file is also refreshed. This secondary action invalidates all current file handles and causes a "stale handle" notification to be returned to any client that attempts to use a previously-mounted file system. Thus, clients must remount any file systems that have been in use when this type of refresh operation is performed.

Allowing Access to Migrated SFS and BFS Files

By default, the CMS SET RECALL setting for the NFS server virtual machine is OFF, which indicates that a reference to data within an SFS or BFS file that is in *migrated status* (that is, has been moved to DFSMS/VM storage) will not cause that data to be implicitly recalled. With respect to the NFS server, the CMS SET RECALL OFF setting causes an NFSERR_IO error to be returned to a client when such a reference is made.

If you choose to change this setting to prevent clients from encountering this error, you must create a server profile exit for the NFS server. Within the NFS server exit, include the SET RECALL ON command in the BEGIN processing section of the exit. In addition, the DTCPARMS file must include an `:exit.` tag that identifies the NFS server profile exit.

Note: When the SET RECALL setting is changed to ON, NFS clients may encounter long delays when requests and references are made for data that is maintained in migrated files.

Managing Data Transfer Operations

Depending on the NFS protocol that is used by a client, it is possible to perform some customization that can affect how data transfer operations are completed by the NFS server.

If the NFSv2 protocol is used by a client, the NFS server uses a maximum of 8192 bytes for READ and WRITE data transfer operations when requests are satisfied. This is a limitation imposed by the NFSv2 protocol specification that applies to both the UDP and TCP transport protocols.

For the NFSv3 protocol, the NFS server uses data buffer values that are defined for the TCP/IP stack to determine its maximum and preferred READ and WRITE data transfer sizes. The UDP values used by the NFS server are based on the LARGEENVELOPEPOOLSIZE `lrg_env_size` value that is specified in the TCP/IP server configuration file. The TCP values used are based on the DATABUFFERPOOLSIZE `buffer_size` value (up to a maximum of 65,536 bytes) that is defined within this same file. For both UDP and TCP, the actual data transfer sizes used by the NFS server may be slightly less than their corresponding TCP/IP server configuration values, due to the inclusion of RPC header information that is required on all data transfers.

When the NFSv3 protocol is in use, NFS clients can determine the maximum and preferred READ and WRITE data transfer sizes that have been established for the NFS server (and which govern its capabilities). This information can then be used by clients to help determine the data transfer sizes they should specify when requests are made of the NFS server.

Note: The maximum number of concurrent NFS client tasks is 64. This is not configurable.

Adjusting TCP/IP Data Buffers

To determine the NFSv3 maximum and preferred data sizes that are currently in use by the NFS server, issue the following SMSG command from an appropriately authorized user ID, such as TCPMAINT:

```
smsg server_id m query config
```

A sample response for this command follows, in which the UDP maximum and preferred READ and WRITE data transfer sizes are each 8784 bytes, while for TCP they are 7784 bytes:

```
NFS server - VM TCP/IP function level 730
Exportonly no, Anonymous yes, Dumpmount yes, Security ESM
PCNFSD yes, time out after 300 seconds
UDP V3 buffer sizes: 8784 read, 8784 write, 8784 preferred.
TCP V3 buffer sizes: 7784 read, 7784 write, 7784 preferred.
Maximum TCP clients: 50.
No NFS clients are currently using TCP.
```

If the data transfer sizes in use are not satisfactory, the following steps can be used to change their values:

1. Stop the TCP/IP server.
2. Modify the appropriate values for the LARGEENVELOPEPOOLSIZE and DATABUFFERPOOLSIZE statements in the TCP/IP server configuration file.

3. Restart the TCP/IP server.
4. Restart the NFS server.
5. Issue the previously cited SMSG command to verify the new data transfer sizes are in effect.

Keep in mind the following when you consider whether to change data transfer size values:

1. Changes to LARGEENVELOPEPOOLSIZE and DATABUFFERPOOLSIZE statement values affect all TCP/IP users, not just the NFS server.
2. Current users of TCP/IP services will be affected when the TCP/IP and NFS servers are stopped and restarted.
3. Additional virtual storage may need to be defined for the TCP/IP server virtual machine when larger envelope or data buffer sizes are specified.

Managing File Handle Operations

File Handle Overview

A *file handle* is a data structure used by NFS to identify a particular file that is to be used. File handles are generated by the NFS server in response to client requests, such as MOUNT and LOOKUP FILE. A client saves this file handle, and returns it when it issues subsequent requests (for example, READ FILE) that pertain to a given file. The data present in a file handle is meaningless to the client system.

For a VM minidisk, file-level control mechanisms do not exist — only disk-level access controls are present. If a minidisk can be mounted, *all* files on that minidisk are accessible to a client. Put another way, the file handle returned for a minidisk MOUNT represents the capability to access *any* file on the mounted disk. By comparison, the VM Shared File System (SFS) and Byte File System (BFS) include inherent file and directory-level control mechanisms. Thus, the file handles returned to clients when these file systems are referenced provide file-level control.

The VMNFS HISTORY File

Because the 32-bytes that comprise a file handle are not adequate for identifying VM files, the NFS server maintains information about the file handles it distributes to clients within a file on its 191 minidisk — the VMNFS HISTORY file. Each file handle generated by the server is associated with record within this file. If a file handle received by the server designates a file that is not immediately known, the appropriate record from this history file is retrieved so that information needed to satisfy a request (such as minidisk link and control block structure information) can be obtained.

Note: Be aware that the VMNFS HISTORY file contains passwords. Therefore, access restrictions appropriate for your installation should be placed on the NFS server 191 minidisk.

Managing the VMNFS HISTORY File

Based on how NFS services are used within your environment (and due to system management operations that may affect the NFS server), actions may at times be necessary to manage certain aspects of the VMNFS HISTORY file.

Accommodating a Large User Population

Within the VMNFS HISTORY file, one record is used for each distinct VM user ID that accesses a minidisk, and one record is used for each distinct VM user ID that accesses one or more SFS or BFS directories in a given file pool. Because of this, the VMNFS HISTORY file may need to be enlarged to accommodate installations that have a large number of users. If this is necessary, the VMNFS HISTORY file must be increased to include additional records, all of which must contain only binary zeros. This can be done by appending the VMNFS HISTORY file that resides on the TCPMAINT 591 minidisk to the active history file present on the NFS server 191 minidisk. For example, assuming that one is logged on to the NFS server user ID and the TCPMAINT 591 minidisk is accessed at file mode E, the command:


```
copyfile vmnfs history e = = a (append
```

will add an additional 640 records to the active file on the NFS server A-disk.

Accommodating System Management and Similar Changes

On rare occasions a file pool administrator may find it necessary to recreate (FILESERV GENERATE) and reload all of the data maintained by a file pool server. However, this action can create problems if NFS clients attempt to use file handles that were created prior to the generation of a file pool, since file handles contain encoded pointers to the objects within a file pool. For example, after a FILESERV GENERATE operation, it is likely that a previously existing file pointer now points to a different file, or even to a directory. For this reason, it is necessary to ensure that all NFS-mounted SFS directories are unmounted before any file pools that contain those directories are (re)generated.

One method of forcing all mounted SFS directories to be unmounted is to stop the NFS server and erase the existing VMNFS HISTORY file. However, this approach also forces any mounted BFS directories or CMS minidisks to be unmounted.

Note: It is recommended that the VMNFS HISTORY file be erased whenever a new TCP/IP Function Level is installed.

Managing Old and Invalid File Handles

Once a file handle has been generated, it has no intrinsic life span, so it is valid indefinitely. However, a simple way of ensuring that file handles have a limited life span is to stop the NFS server and erase the active VMNFS HISTORY file at regular intervals. This forces clients to remount referenced file systems in order to receive new, valid file handles.

Note:

1. When the VMNFS HISTORY file is erased, the VMNFS TRANSLAT file is also refreshed when the NFS server is again initialized.
2. There is no security exposure due to attempts to use old file handles; unauthorized access to data continues to be restricted in such cases.

Using Additional Security Capabilities

File Handle Encryption (NFSHCIP ASSEMBLE File)

To guard against the possibility that a client has modified or devised a file handle to gain unauthorized access to a file, the NFS server has the ability to encrypt all file handles. When enabled, each file handle that is generated by the NFS server in response to a client request is encrypted before it is provided to the client. When a file handle is returned to the NFS server, it is decoded to obtain the original structure that identifies the associated file. If the server detects a tampered encryption, the decoding process generates an invalid file handle and rejects the accompanying request.

Note: File data is not encrypted by the NFS protocol, and the NFS server does not have such capabilities. If data encryption is necessary for your environment, this must be performed independent of NFS server operations.

The NFSHCIP ASSEMBLE file (Network File System File Handle Cryptographic Interface Program) can be used to invoke a cipher routine in order to encode and decode a file handle. As supplied, this file will default to invoking a cipher program named IPSASM, for which a corresponding IPSASM TEXT file must exist. A different cryptographic routine can be used in place of the IPSASM routine, though this may require minor changes to NFSHCIP; for more information, see [“Source Code Modifications” on page 342](#).

Minidisk Link Monitoring (NFSBADPW C File)

When a client *minidisk* mount request is processed, an internal NFS server routine (NFSBADPW) is called after an attempt has been made to link the requested minidisk; this routine is called regardless of

whether this link attempt succeeds or fails. As supplied with TCP/IP, this routine simply issues a message which contains details about the link failure to the NFS server console.

If additional action is desired or necessary when a link failure occurs (such as logging data in a disk file or informing the system operator or another user of a failure), the NFSBADPW routine can be updated or replaced, through modification of the NFSBADPW C program source file.

Source Code Modifications

To make use of the security capabilities described in the previous sections, source code modifications are necessary which then require the NFS server program (VMNFS MODULE) to be rebuilt.

If file handle encryption is to be used by the NFS server, the following modifications are required:

1. Modify the NFSHCIP ASSEMBLE file to employ suitable calling conventions for the encryption program that is to be used.
2. If necessary, modify the TCPBLC91 EXEC (the VMSES/E build list that is used to build the VMNFS MODULE). Changes to this build list will be required if the cipher routine text file (supplied with the encryption program in use) is not named, or cannot be renamed to, IPSASM TEXT. In such a case, the IPSASM TEXT entry within the TCPBLC91 EXEC must be changed to reflect the proper name.

If the NFSBADPW C file is to be updated or replaced, this file must be suitably modified and compiled, and the VMNFS MODULE must then be rebuilt.

For more information about making these modifications and rebuilding the VMNFS MODULE within the VMSES/E environment, see the *Program Directory for TCP/IP for z/VM*, beginning with the appendix that discusses VMNFS code modifications.

Dynamic Server Operation

The VM Special Message Facility (SMSG) command provides an interactive interface to the NFS server to:

- modify NFS server configuration attributes
- obtain various types of information about server operations and client mounts
- instruct the server to perform certain actions.

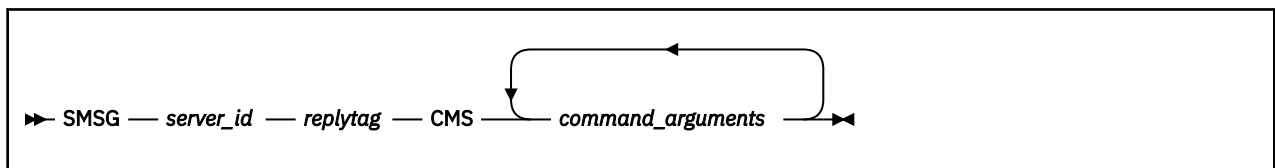
Note: The NFS server SMSG authorization exit can be used to control the acceptance and processing of SMSG commands issued to the NFS server. For more information, see [“SMSG Authorization Exit” on page 337](#).

For detailed information about user-oriented SMSG commands that are supported by the NFS server, see the section titled *SMSG Interface to VMNFS* in the *TCP/IP User's Guide*.

SMSG Interface to the NFS Server

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

SMSG CMS Command



Purpose

Use the SMSG CMS command to direct the NFS server to pass commands to CMS for execution. Note that all commands are processed under control of the VMNFS CMS command exit.

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.

command_arguments

Commands that are to be processed by CMS.

Usage Notes

- 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.

SMSG REFRESH CONFIG Command

➡ SMSG — *server_id* — *replytag* — Refresh CONFIG ➡

Purpose

Use the SMSG Refresh CONFIG command to direct the NFS server to replace existing configuration information with that defined by current definitions in the NFS server configuration file (VMNFS CONFIG).

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.

Usage Notes

- 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.
- Except for the MAXTCPUSERS statement, all records within the NFS configuration file are used to update NFS server configuration parameters. If a supported configuration statement is not present within this file, the default for that statement is used. For example, if the PCNFSD statement is omitted, the default value of YES is used.
- The NFS server configuration file typically resides on the TCPMAINT 198 minidisk. For changes to this file to become effective, the NFS server must reaccess this minidisk prior to receipt of an SMSG REFRESH CONFIG command. A reaccess of the TCPMAINT 198 minidisk can be accomplished by issuing an appropriate SMSG CMS command to the NFS server, such as:

```
smsg server_id m access 198 d
```

- If changes are made to the NFS configuration file and the SMSG REFRESH CONFIG command is not used, those changes become effective when the NFS server is again initialized.

SMSG TWRITE Command

```
►► SMSG — server_id — replytag — TWRITE ►►
```

Purpose

Use the SMSG TWRITE command to cause trace tables to be written to file TRACEV FILE on the server's A-disk.

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.

Usage Notes

- 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.

Chapter 10. Configuring the Portmapper Server

The Portmapper (PORTMAP) server application is used to map program numbers and various numbers to RPC programs that request information. To configure the Portmapper server virtual machine, you must perform the following steps:

Portmapper Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the Portmapper server.
3. Verify Portmapper services.

Step 1: Update PROFILE TCPIP

Include the Portmapper server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The Portmapper server is then automatically started when TCPIP is initialized. The IBM default user ID for this server is PORTMAP. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
PORTMAP 0
```

Note: If your system is using the Network File System (NFS) server, the Portmapper server must be started before the NFS server. If the NFS server does not receive a response from a Portmapper request, it waits a period of time for the Portmapper server to complete initialization, then tries again.

The Portmapper server requires that ports UDP 111 and TCP 111 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
111 UDP PORTMAP      ; Portmapper Server
111 TCP PORTMAP      ; Portmapper Server
```

Step 2: Update the DTCPARMS File

When the Portmapper server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the Portmapper server are:

```
:Nick.PORTMAP
:Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Note: You should modify the DTCPARMS file for the Portmapper server if you change the user ID of the virtual machine that receives the Portmapper console output.

PORTMAP Command Operands (:Parms. Parameters)

Portmapper services are initiated using the PORTMAP command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with a Portmapper server definition. For more information about this command and its operands, see [“PORTMAP Command Syntax”](#) on page 348.

PORTMAP Command Syntax



Operands

-d

Turns on debug tracing. Trace information is directed to the Portmapper server console.

Step 3: Verify Portmapper Services

To verify that Portmapper services are available:

1. If necessary, link and access the TCPMAINT 592 client code disk.
2. Issue the following command:

```
rpcinfo -p loopback
```

At a minimum, the response to this command should include the following:

program	vers	proto	port	
100000	2	upd	111	portmapper
100000	2	tcp	111	portmapper

Chapter 11. Configuring the REXEC Server

The REXEC server implements the Remote Execution Command protocol (REXEC). To configure the REXEC server virtual machine, you must perform the following steps:

REXEC Server Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the DTCPARMS file for the REXEC server. 3. Define additional REXEC agent virtual machines. (Optional)

Step 1: Update PROFILE TCPIP

Include the REXEC server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The REXEC server is then automatically started when TCPIP is initialized. The IBM default user ID for this server is REXECD. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
  REXECD    0
```

The REXEC server requires that TCP port 512 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
  512 TCP REXECD    ; REXEC Server
  514 TCP REXECD    ; REXEC Server
```

To allow the REXEC server to manage its agent virtual machines, it must be included in the OBEY list in PROFILE TCPIP. Verify that the following statements are added to PROFILE TCPIP:

```
OBEY
  REXECD
ENDOBEY
```

For more information about the OBEY list, see [“OBEY Statement” on page 597](#).

Step 2: Update the DTCPARMS File

When the REXEC server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the REXEC server are:

```
:Nick.REXECD
:Parms.
:Anonymous.
:ESM_Enable.
:ESM_Validate.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,” on page 33](#).

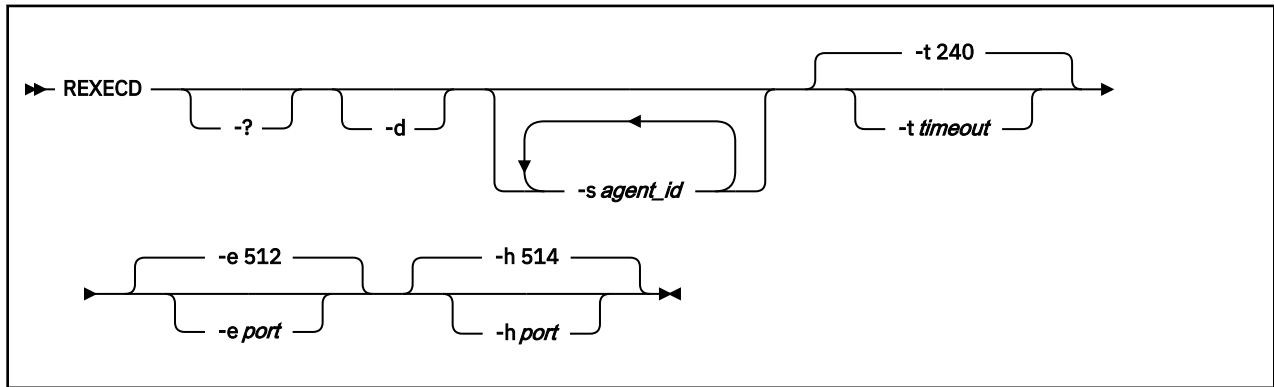
Note: You should modify the DTCPARMS file for the REXEC server if you:

- Run the server with an External Security Manager (ESM), such as RACF.
- Choose to enable anonymous rexec capabilities.
- Override default command parameters for this server.

REXECD Command Operands (:Parms. Parameters)

REXEC services are initiated using the REXECD command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with an REXEC server definition. For more information about this command and its operands, see [“REXECD Command Syntax” on page 350](#).

REXECD Command Syntax



Operands

-?

Displays all options supported by the REXECD command.

-d

Turns on debug tracing.

-s *agent_id*

Identifies the *agent_id* virtual machine as a member of the anonymous client agent server pool. Repeat this parameter to identify any additional agent machines that will be used. It is recommended that you do not specify this parameter using the :Parms. tag, but instead specify :Anonymous.YES in the DTCPARMS file.

All agents must have :Class.rexec_agent and :For.rexecd (the user ID of the REXEC server) specified in the DTCPARMS file. A default agent virtual machine, RXAGENT1, is defined in this file.

See [“Anonymous REXEC Client Processing” on page 351](#) for more information about how agents are used.

-t *timeout*

Sets idle timeout. This option specifies the time (in seconds) in which a connection closes if there is no activity. The default timeout is 240 seconds.

-e *port*

Sets the REXEC port. The default REXEC port is 512.

-h *port*

Sets the RSH port. The default RSH port is 514.

Step 3: Define Additional Anonymous REXEC Agent Virtual Machines (Optional)

To provide support for *anonymous* REXEC client requests, at least one REXEC *agent* virtual machine must be defined for your installation. The default TCP/IP installation environment includes one such agent virtual machine, named RXAGENT1.

Because anonymous REXEC requests are processed using only these agent virtual machines, the REXEC server can respond to such requests more readily when more than one agent machine is available.

For any additional REXEC agent machines that you define, it is recommended that you:

- maintain the RXAGENT n naming convention
- model your CP directory entries after that supplied for the RXAGENT1 virtual machine.

For more information about duplicating existing servers, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33. If necessary, for specific DASD storage and user ID requirements that may be applicable to these virtual machines, consult *Program Directory for TCP/IP for z/VM*.

Step 4: Establish REXEC Server Machine Authorizations

The CP directory entry for the REXECD server must include an OPTION DIAG 88 statement.

The REXECD server can use an external security manager (ESM) to authenticate FTP clients and to control access to z/VM resources. To use an ESM, specify :ESM_Enable .Yes in the DTCPARMS file. For more information, see [Appendix A, “Using TCP/IP with an External Security Manager,”](#) on page 685.

Using an External Security Manager

The REXEC server can be configured such that client authentication will be under the control of an external security manager (ESM), such as RACF. For more information on using an ESM, see [Appendix A, “Using TCP/IP with an External Security Manager,”](#) on page 685.

Additional REXEC Considerations

The sections that follow provide information about operational and processing characteristics inherent to the z/VM REXEC server implementation, which may help with the administration and use of REXEC services within a given environment.

How the REXEC Server Uses Secondary Virtual Machines

The REXEC server uses secondary virtual machines (agents) to execute commands passed from each ANONYMOUS or GUEST client. This allows the REXEC server to better service multiple REXEC requests, as the multiple agents accessible to the REXEC server help simulate a multitasking environment. Additionally, a user's own virtual machine can be used to process REXEC commands; when this is done, the user machine is managed somewhat differently than an agent machine.

Anonymous REXEC Client Processing

If :Anonymous .YES is specified in the DTCPARMS file, the REXEC server will maintain a "pool" of *agent* virtual machines to handle REXEC clients that log in as "anonymous" or "guest". When the REXEC server is started, there is a short delay (approximately 30 seconds) during which all agents in the pool are made ready; during this time, REXEC clients that attempt to use anonymous services will receive an indication that no agents are available. Once the agents are ready (logged on), the REXEC server can accept anonymous REXEC requests.

When an anonymous REXEC command is received, the command is sent to an available agent. That agent issues the command and returns the command response to the REXEC server, which then sends the response back to the REXEC client. When the transaction is complete, the agent is returned to the pool of available agents.

Agents are reinitialized by the REXEC server when:

- the REXEC server is itself reinitialized.
- an agent is determined to be inoperable, due to a problem in processing a command (that is, it does not return command output within the defined timeout period).

User's Own Virtual Machines

When a user's own virtual machine is used to execute an REXEC-supplied command, that machine is autologged by the REXEC server (by the XAUTOLOG command) after the supplied user and password have been authenticated. The given command is then processed in a manner similar to those supplied to an anonymous agent machine. However, after the command has completed, the user's virtual machine is logged off, rather than returned to the anonymous agent pool.

When a special format of user is specified, this indicates that the client wants to use LOGONBY privileges. The format is:

user_id/BY/byuser_id or
user_id.BY.byuser_id

The password for *byuser_id* is used for LOGON authorization checking for the user ID specified in *user_id*. The user ID specified in *byuser_id* must be listed in a LOGONBY statement in the directory entry in order for *user_id* to use this format.

Usage Notes

- All agent virtual machines (RXAGENTn machines) must use the REXEC server's PROFILE EXEC to function properly.
- Because the REXEC server and the agent or user virtual machines communicate using CP messages, the Single Console Image Facility (SCIF) may not be used to monitor the REXEC server, agent, or user consoles.

For more information about using REXEC commands with agent and user machines, and restrictions on their use, see the [*z/VM: TCP/IP User's Guide*](#), specifically the chapter that deals with using the remote execution protocol.

Chapter 12. Configuring the RSCS Print Server

The RSCS server may be chosen to provide an enhanced level of TCP/IP print support, including LPR and LPD.

Using RSCS provides end users with the following:

- Deferred (asynchronous) printing. The user's CMS session is not left idle waiting to print a file.
- TN3270E printer sessions, providing workstation print capabilities when LPD is not installed or available on the workstation.
- Enhanced error recovery. RSCS will periodically attempt to retransmit a print file in the event of a TCP/IP network or remote printer failure.
- Enhanced QUERY capabilities. RSCS commands are available to the end user to determine the status of their print request, complementing the existing LPQ and LPRM commands.
- The ability for workstation users to use LPR to print on any printer owned by the VM system. When a license for RSCS Networking for z/VM is acquired, this capability is extended to any printer or user anywhere in the RSCS network.

In addition to the advantages provided to end users, it enables the integration of TCP/IP printing into an existing RSCS printer network and its associated management.

Configuring a TN3270E Printer

The TN3270E protocol supported by TCP/IP provides for the creation of 3270 printer sessions in addition to traditional display sessions. To make this capability available, the following steps must be performed:

1. An arbitrary name, called the "LU name" must be defined in the TN3270E control statement in the PROFILE TCPIP file. This statement assigns a virtual line address to the printer session.
2. The TN3270E Printer Management exit must be enabled by the TN3270EEXIT parameter of the INTERNALCLIENTPARMS statement in the PROFILE TCPIP file. See ["INTERNALCLIENTPARMS Statement" on page 584](#).
3. An RSCS TN3270E link must be defined that has a line address that matches the line address defined in step 1. While not required, you may find administration and problem determination easier if the LU name is the same as the associated RSCS link name. The sample TN3270E Printer Management exit assumes the LU name and RSCS link names match.
4. The user must have the "LU name" configured in their TN3270E-capable emulator.

Configuring an RSCS LPR Link

To configure an RSCS LPR link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. Different options must be used depending on whether the printer is PostScript or non-PostScript.

RSCSTCP CONFIG Configuration File

The RSCSTCP CONFIG configuration file contains statements you can use to define your RSCS network. This file is read during initialization of the RSCS virtual machine. If this file is not found, RSCS initialization will fail. This file is the main configuration file for the RSCS server and will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. It allows you to specify:

- The name of your local RSCS node if you choose it to be different from the system network ID
- LPR-type links for use with non-PostScript printers
- LPR-type links for use with PostScript printers
- LPD-type links for use as a local daemon

- TN3270E-type links for use with telnet attached 3287–1 printers
- UFT-type link for use as a local asynchronous UFT client

Two LPR links have been defined in the RSCSTCP CONFIG sample RSCS configuration file on TCPMAINT's 198 minidisk with the following *linkid*:

- LPR, which is to be used with non-PostScript printers
- LPRP, which is to be used with PostScript printers

Configuring a Non-PostScript Printer

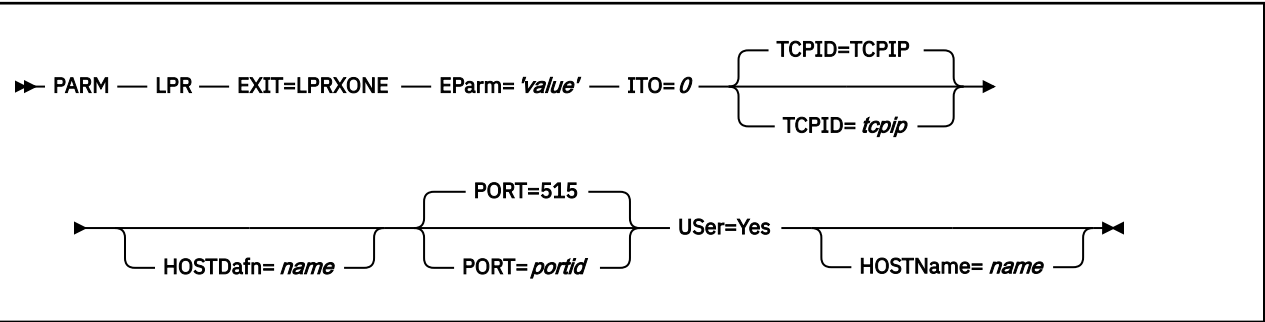
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPR link for use with non-PostScript printers.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

```
➤➤ LINKDEFine — LPR — AST — FOrn * — TYPE LPR ➤➤
```

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPR keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPR link that connects your local RSCS server to a remote line printer daemon (LPD) in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.



Parameter Description

EParm='value'
value is a character string up to 239 bytes in length and enclosed in single quotation marks (' '). For additional information see [“Available EPARMS for Non-PostScript Printers”](#) on page 355.

PORT=portid
Specifies the port number to use when connecting to a remote host. The default is **well known** port 515.

TCPID=tcPIP
Specifies the name of the TCP/IP server; the default name is TCPIP.

HOSTDafn=name
Specifies a 1- to 8-character name which should be used as the host name portion of the control and data file names. A control or data file name is used within the 'Receive control file' and 'Receive data file' subcommands of the daemon 'Receive job' command. The name should start with "cfa" (control file) or "dfa" (data file), followed by a three-digit job number, followed by the host name that has constructed the control/data file, as described in RFC 1179. RSCS defaults to using the link name for the host name portion of the file name if this parameter is not specified. Some daemons will not accept any name, other than the host name. This parameter allows you to specify the host name.

HOSTName=name
Specifies a 1- to 200-character fully qualified name of the remote host.

Available EPARMS for Non-PostScript Printers

The RSCS LPRXONE exit routine performs function for a non-PostScript printer. It also performs simple translation of data to ASCII. The following EPARM values can be used to configure the LPRXONE exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Parameter Description

FF=

Determines how printer form-feeds are performed.

TOP

Form-feed is sent in the front of the file

Bottom

Form-feed is sent after the file; this is the default.

None

A form-feed is not sent.

Filter=f

Specifies the printer filter used in the control file sent to the daemon if a filter is not specified by the user when the LPR command is issued; the default is f. One EBCDIC character is passed. If it is uppercase alphabetic, it will be translated to lowercase. No other validation is performed. This maintains consistency with RFC1179.

Prefix=hex_string

Optionally specifies a hexadecimal string to be sent in front of each file if a prefix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to pass printer specific setup values. Up to 200 bytes of data can be specified. By default, a prefix string is not sent with each file.

Sep=

Indicates if a separator page will be printed for each file.

Yes

Prints a separator page; this is the default. The origin user ID and node ID and distribution information are printed in large characters; other file information is printed in small characters in the Times-Bold font.

No

Does not print a separator page

Host

Sends the L control file record to request that the remote host produce the separator page.

SUFFIX= hex_string

Optionally specifies a hexadecimal string to be sent after each file if a suffix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to reset the printer upon print completion. Up to 200 bytes of data can be specified. By default, a suffix string is not sent with each file.

Config=LPR

Causes the LPRXONE exit to read the RSCSLPR CONFIG sample configuration file located on TCPMAINT's 198 minidisk. The configuration file can contain translation tables to override the tables used within the exit.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

ASCII=

Provide a table for translating ASCII control characters, overriding the default used by the exit. LPRXONE uses this translation table when files are already in ASCII and the user ID field of the

TAG is set to 'ASCIIC'. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple ASCII= records to replace the 256-byte translation table.

DOMAINNAME=

Specifies a domain name, up to 255 characters, to be appended after the host name of the 'H' control file record. A period (.) will be inserted between the host name and domain name. This record can be used to add a domain name after the host name, which by default is the node name where the file originated or as specified in the HOSTNAME= record.

HOSTNAME=

Used to specify a host name, up to 255 characters, for the 'H' control file record, overriding the default, which is the node name where the file originated.

TOASCII=

Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCII= records to replace the 256-byte translation table.

TOASCIIC=

Provide a table for EBCDIC to ASCII translation of the LPR control file, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCIIC= records to replace the 256-byte translation table.

USERNAME=

Specifies a user name, up to 32 characters, for the 'P' control file record, overriding the default name used by the exit, which is the user name of the file originator. This record can be used to cause all error messages to be sent to a central location.

A sample EPARM follows:

```
EPARM='C=LPR S=N FF=N'
```

Configuring a PostScript Printer

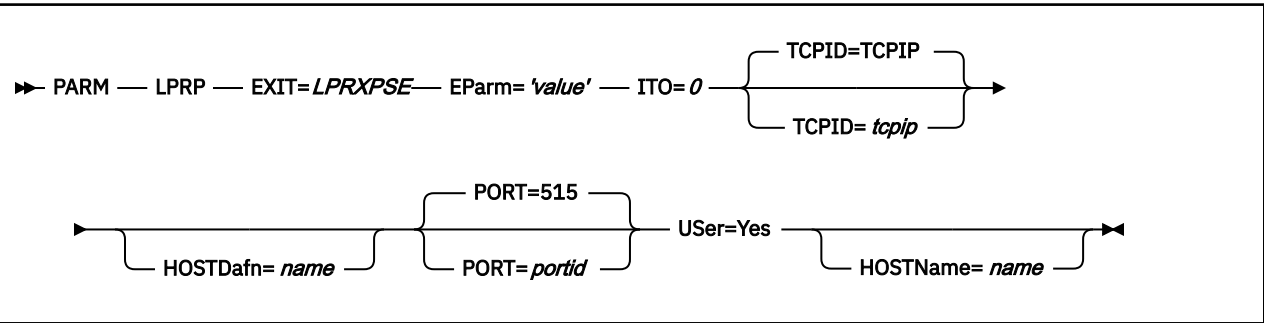
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPR link for use with PostScript printers.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.



Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPRP keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPR link that connects your local RSCS server to a remote line printer daemon (LPD) in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.



Parameter Description

EParm='value'

value is a character string up to 239 bytes in length and enclosed in single quotation marks (' '). For additional information see [“Available EPARMS for PostScript Printers”](#) on page 357.

PORT=portid

Specifies the port number to use when connecting to a remote host. The default is **well known** port 515

TCPID=tcPIP

Specifies the name of the TCP/IP server; the default name is TCPIP.

HOSTDafn=name

Specifies a 1- to 8-character name that should be used as the host name portion of the control and data file names. A control or data file name is used within the 'Receive control file' and 'Receive data file' subcommands of the daemon 'Receive job' command. The name should start with "cfa" (control file) or "dfa" (data file), followed by a three-digit job number, followed by the host name that has constructed the control/data file, as described in RFC 1179. RSCS defaults to using the link name for the host name portion of the file name if this parameter is not specified. Some daemons will not accept any name, other than the host name. This parameter allows you to specify the host name.

HOSTName=name

Specifies a 1-to 200-character fully qualified name of the remote host.

Available EPARMS for PostScript Printers

The RSCS LPRXPSE exit routine performs functions for a PostScript printer. This exit routine assumes that the remote printer is PostScript only, or that it will switch into PostScript mode when it receives a "%!PS" string following an EOT character. It also performs simple translation of data to ASCII. The following EPARM values can be used to configure the LPRXPSE exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Ehandler=

Determines if a PostScript error handler will be downloaded to the printer the first time a file is sent to the printer after the link is started. This error handler enables any errors to be printed, so the information will not be lost.

Yes

The error handler is downloaded; this is the default.

No

The error handler is not downloaded.

EOT=**Yes**

EOT characters will be inserted after the separator page, data file, and trailer page; this is the default.

No

EOT characters will not be inserted.

Filter=f

Specifies the printer filter used in the control file sent to the daemon if a filter is not specified by the user when the LPR command is issued; the default is f. One EBCDIC character is passed. If it is uppercase alphabetic, it will be translated to lowercase. No other validation is performed. This maintains consistency with RFC1179.

Form=OrFnFsLs

Specifies the default form to use when printing plain text files and when a form has not been specified on the LPR command.

The following can be used for *OrFnFsLs* and shows the defaults.

Or

File orientation:

PO

Portrait (default)

LA

Landscape

Fn

Font name code

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 size, 04 thru 99; the default is 11 for portrait and 10 for landscape orientation

Ls

Additional leading size, 0.0 - 9.9, added to font size to give leading; the default is 09 for portrait and 12 for landscape

Prefix= *hex_string*

Optionally specifies a hexadecimal string to be sent in front of each file if a prefix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to pass printer specific setup values. Up to 200 bytes of data can be specified. By default, a prefix string is not sent with each file.

Sep=

Indicates if a separator page will be printed for each file.

Yes

Prints a separator page; this is the default. The origin user ID and node ID and distribution information are printed in large characters; other file information is printed in small characters in the Times-Bold font.

No

Does not print a separator page

Host

Sends the L control file record to request that the remote host produce the separator page.

SUFFIX= *hex_string*

Optionally specifies a hexadecimal string to be sent after each file if a suffix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to reset the printer upon print completion. Up to 200 bytes of data can be specified. By default, a suffix string is not sent with each file.

Trailer=

Indicates if a trailer page will be printed

Yes

Prints a trailer page after the file. It is identical to the header page with the addition of a count of the bytes in the file.

No

Trailer page is not printed; this is the default.

Config=LPRP

Causes the LPRXPSE exit to read the RSCSLPRP CONFIG sample configuration file located on TCPMAINT's 198 minidisk.

This configuration file is used to supply the following:

- To override the default translate table.
- To override the PostScript program sent to the printer when printing plain text files.
- Additional font names used when printing plain text files.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

DOMAINNAME=

Specifies a domain name, up to 255 characters, to be appended after the host name of the 'H' control file record. A period (.) will be inserted between the host name and domain name. This record can be used to add a domain name after the host name, which by default is the node name where the file originated or as specified in the HOSTNAME= record.

HOSTNAME=

Used to specify a host name, up to 255 characters, for the 'H' control file record, overriding the default, which is the node name where the file originated.

FONT=

Provides a 2-character font abbreviation followed by a 32-character font name. There should be no blanks between the abbreviation and full font name. Multiple records can be provided for supplying as many additional fonts as required. The abbreviation should be unique on each FONT= record. The fonts must be installed on the printer. The current available font names are:

2-Character**32-Character****Abbreviation****Font Name****CB**

Courier-Bold

CI

Courier-Oblique

CP

Courier (exit 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

PSCRIPT=

Provide a replacement PostScript program to be used when printing a plain text file. The PostScript program must be enclosed within quotation marks. Anything after the ending quotation mark will be ignored allowing for comments.

For example:

```
PSCRIPT='this is line one'  comment for line one
PSCRIPT='this is line two'
```

Multiple PSCRIPT= records can be provided in order to supply the entire program. LPRXPSE will add a carriage return (X'0A') after each record, and will translate the record from EBCDIC to ASCII.

Note: When replacing the PostScript program, the ability to tailor the file orientation, font name, font size, and additional leading size through a FORM is lost. The supplied PostScript program must define all of these attributes.

TOASCII=

Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCII= records to replace the 256-byte translation table.

TOASCIIIC=

Provide a table for EBCDIC to ASCII translation of the LPR control file, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCIIIC= records to replace the 256-byte translation table.

USERNAME=

Specifies a user name, up to 32 characters, for the 'P' control file record, overriding the default name used by the exit, which is the user name of the file originator. This record can be used to cause all error messages to be sent to a central location.

The following is a sample EPARM:

```
EPARM='C=LPRP S=N T=N'
```

Form Parameter of LPR Command When Printing to PostScript

When printing to PostScript printers, you can also specify the following values on the FORM=*OrFnFsLs* operand of the LPR command.

Or

File orientation:

PO

Portrait

LA

Landscape

Fn

Font name code

CB

Courier-Bold

CI

Courier-Oblique

CP

Courier

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 size, 04 thru 99.

Ls

Additional leading size, 0.0 - 9.9, added to font size to give leading.

Configuring an RSCS LPD Link

The RSCSLPD CONFIG configuration file contains statements you can use to customize an RSCS LPD-type link. This file is read during link initialization. If this file is not found, the link initialization will fail. This file will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. It allows you to specify:

- A replacement ASCII to EBCDIC translation table
- A replacement EBCDIC to ASCII translation table
- Definitions to be used while receiving a file for a particular printer queue name. The definitions which may be defined are
 - a queue name

- logical record length
- number of lines per page
- spool file class
- spool file form
- job name
- PSF destination
- whether or not to paginate
- whether or not to translate
- destination user ID
- destination node ID
- TCPXLBIN translation table to use for ASCII to EBCDIC translations

To configure an RSCS LPD link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. One LPD link has been defined in the RSCSTCP CONFIG sample RSCS configuration file on TCPMAINT's 198 minidisk.

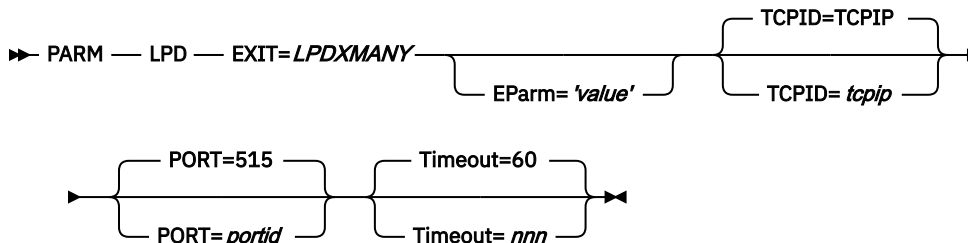
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPD link.

►► LINKDEFine — LPD — TYPE LPD ◄◄

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPD keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPD link that will act as a gateway accepting print data streams from the TCP/IP environment.



Parameter Description

EParm=*'value'*

value is a character string up to 239 bytes in length and enclosed in single quotation marks (' '). For additional information see [“Available EPARMs for LPD Links”](#) on page 363.

TCPID=*tcPIP*

Specifies the name of the TCPIP server; The default name is TCPIP.

PORT=*portid*

Specifies the port number to use when listening on the local host. The default is **well known** port 515.

Timeout=*nnn*

Specifies the amount of time in seconds RSCS will wait for data when receiving from a TCP/IP line print router prior to breaking the socket connection; the default is 60. After the LPD link driver breaks this connection, it will remain operational waiting for another line print router to connect.

Available EPARMs for LPD Links

The RSCS LPDXMANY exit routine performs functions to support a single LPD printer queue. Multiple LPD link drivers can be defined using this exit. It also performs simple translation of data to EBCDIC.

The spool file created will be of type VAFP. A record width up to 1280 characters will be supported. Any records received with a width greater than 1280 characters will be split into multiple records. The CMS receive command will create a file with a record length of 204. Any data in a record past 204 will be truncated. CP will also truncate the data with a record length greater than 204 if the spool file is destined to a CP system printer.

Printer job commands received from the line print router contain a queue name of the form *userid@nodeid* or *userid%nodeid*. The following examples show how the LPDXMANY routine parses this queue name, and how the provided information affects a created spool file.

KERRY@MAINE

Will set the node ID to MAINE and the user ID to KERRY. The file will be sent to user KERRY at node MAINE.

KERRY%MAINE

Will set the node ID to MAINE and the user ID to KERRY. The file will be sent to user KERRY at node MAINE.

KERRY@

Will set the node ID to the local node name and the user ID to KERRY. The file will be spooled to user KERRY on the local system.

KERRY%

Will set the node ID to the local node name and the user ID to KERRY. The file will be spooled to user KERRY on the local system.

@LASER1

Will set the node ID to LASER1 and the user ID to SYSTEM causing the file to be sent to the network node LASER1.

%LASER2

Will set the node ID to LASER2 and the user ID to SYSTEM causing the file to be sent to the network node LASER2.

LASER3

Will set the node ID to LASER3 and the user ID to SYSTEM causing the file to be sent to the network node LASER3.

Note: The LPDXMANY routine will limit the length of the user ID and node ID fields to 8 characters. Any extra data in those operand fields will be discarded.

The following EPARM values can be used to configure the LPDXMANY exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Parameter	Description

Config=LPD

Causes the LPDXMANY exit to read the RSCSLPD CONFIG sample configuration file located on TCPMAINT's 198 minidisk. This configuration file is used to supply the following:

- Translation table to override the one used by the exit.
- Supply overrides for processing when a file is received from a remote LPR command based on the printer queue name.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

TOASCII=

Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCII= records to replace the 256-byte translation table.

TOEBCDIC=

Provide a table for ASCII to EBCDIC translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOEBCDIC= records to replace the 256-byte translation table.

queuenam

Provide the ability to override defaults used by LPDXMANY on a printer queue name basis when receiving a file from a remote host. Multiple, unique, printer queue name records can be specified.

Format of Printer Queue Name Records

The layout of the printer queue name records is as follows:

- Each queue name option is separated by one or more blanks.
- The parameters are not column dependent.
- One record per line, continuation is not supported.
- Each parameter IS position dependent.
- An * can be used in a position (other than for the printer queue name) to tell LPDXMANY to use the existing default.
- The printer queue name will be parsed into user ID and node ID as follows:

<userid@>nodeid

<userid%>nodeid

nodeid

user ID will be set to SYSTEM.

@nodeid

user ID will be set to SYSTEM.

%nodeid

user ID will be set to SYSTEM.

userid@

node ID will be set to local node name.

userid%

node ID will be set to local node name.

NONEOFTHEABOVE

node ID and user ID MUST be set within record.

The user ID and node ID parsed from the printer queue name can be overridden within the record. If both are overridden, the printer queue name is not parsed. If neither are overridden, the printer queue name is parsed into user ID and node ID which must be valid (either of which can still be overridden within).

```
queuenam  ppos lpage class forms jobn dest
pagination translation userid nodeid tcpxlbin
DEFAULT ppos lpage class forms jobn dest
pagination translation userid nodeid tcpxlbin
```

Parameter**Description*****queuenam***

is a printer queue name up to 32 characters. A *queuenam* of DEFAULT can be used to define parameters for any printer queue not defined by a configuration file record. When a printer queue name arrives, LPDXMANY first looks for a configuration record of that name. If this is not found,

LPDXMANY will look for a configuration record using DEFAULT; if this is not found, it will use existing LPDXMANY defaults.

ppos

is the logical record length 1-1280; the default is 255.

lpage

is the number of lines per page 1-99; the default is 66.

class

is the 1-character spool file class; the default is blank.

forms

is the 1- to 8-character spool file form; the default is blank.

jobn

is the 1- to 8-character job name; the default is SYSTEM.

dest

is the 1- to 8-character PSF destination; the default is blank. When using a PSF destination, LPDXMANY will set the user ID field of the tag to *SYSTEM*.

pagination

is either PAGE or NOPAGE; the default is NOPAGE. This parameter determines how pagination is performed:

PAGE will cause LPDXMANY to always paginate, regardless of the print filter.

NOPAGE will cause LPDXMANY to paginate only as defined by the control file filters 'f' or 'p'. To be effective, the control file must be received prior to the data file to be printed.

translation

is either ASISCC, NOTRAN or TRAN; the default is TRAN. Translation determines if the data file is translated prior to spooling.

userid

is the 1-to 8-character user ID that will receive the spooled file.

nodeid

is the one- to eight-character destination

tcpxlb

is the file name of a TCP/IP TCPXLBIN translate table to be used for this printer queue name.

QUEUENAME examples:

```
LPR@NODEONE 1280 50 * STDN * * * ASISCC ASCII
DEFAULT 255 66 * * SYSTEM * NOPAGE TRAN *
```

The first example would allow a file to be received without translation, and spooled to an LPR link. ASCII tells the LPRXONE exits that the file is already in ASCII and to send it unaltered to the remote daemon.

The second example would allow a file to be spooled to the system printer.

The following is a sample EPARM:

```
EPARM='C=LPD'
```

Configuring an RSCS TN3270E Printer Link

To configure an RSCS TN3270E printer link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. You must define one TN3270E printer link for each printer LUNAME defined within the TCPIP configuration file.

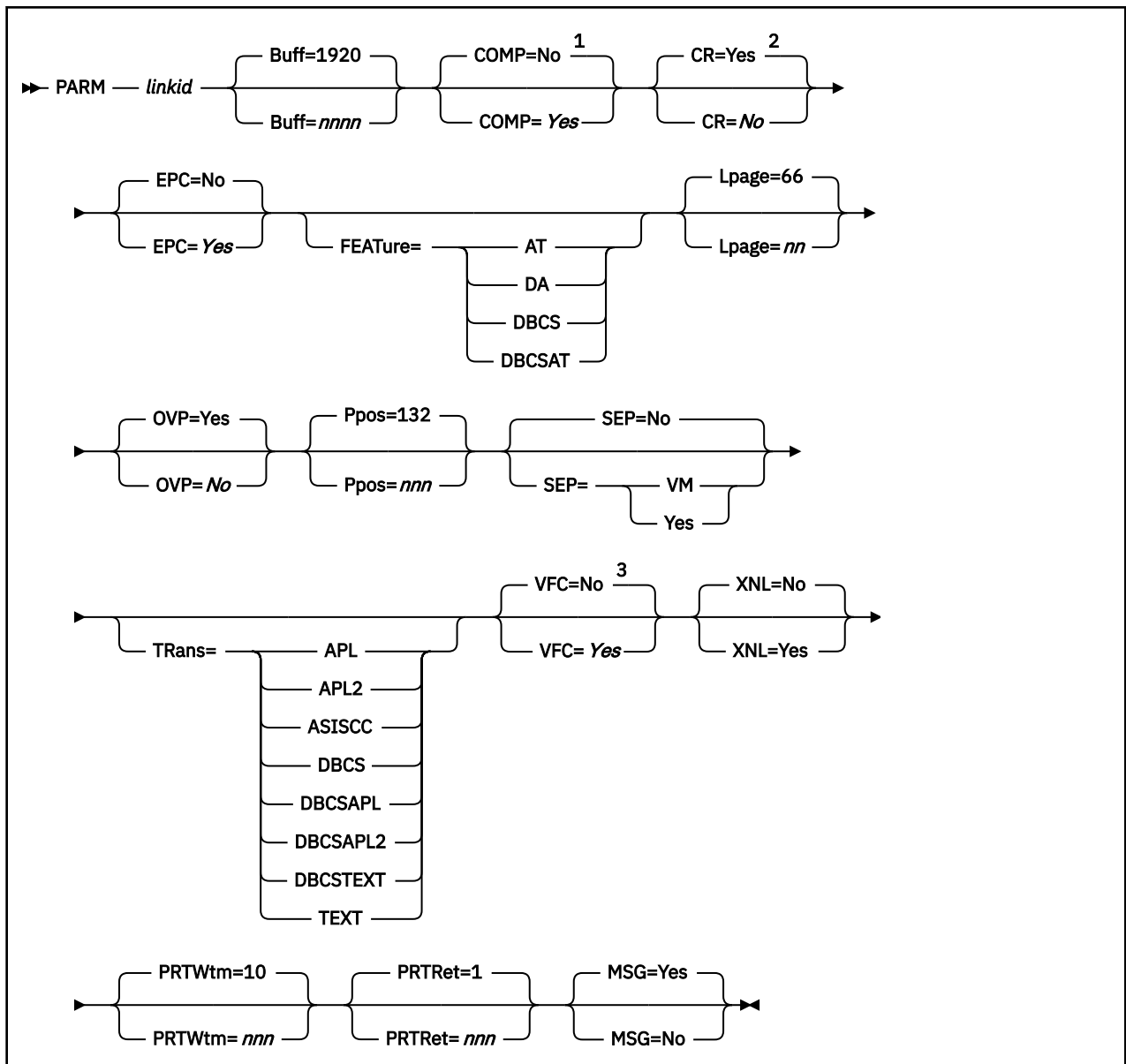
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an TN3270E printer link. The *linkid* may match the LUNAME; however, the line address must match the virtual address defined in the TCPIP configuration file.

```
➤ LINKDEFINE — linkid — AST — LINE vaddr — TYPE TN3270E ➤
```

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

You can specify multiple LINKDEFINE statements for the same link. If you specify more than one LINKDEFINE statement with the same *linkid*, RSCS uses the attribute from the last LINKDEFINE statement.



Notes:

- ¹ Specify NO for locally attached printers.
- ² CR has no meaning unless VFC=YES is also specified.
- ³ Generally it is recommended to specify VFC=YES.

linkid

Is the 1- to 8-character name of the link that connects your local RSCS system to a TN3270E attached printer. Before you can specify a PARM statement for a link, you must specify a LINKDEFINE (or LINK) statement with the same link ID.

Buff=1920**Buff=nnnn**

Is the buffer size of the 3287-1 printer. The following buffer sizes are valid: 480, 960, 1920, 2560, 3440, and 3564. The correct buffer size has to be specified to match the buffer size on the printer being used. If this keyword is omitted, the buffer size defaults to 1920.

COMP=Yes**COMP=No**

Specifies that the link will perform blank compression. If omitted, the default is to not compress.

CR=Yes**CR=No**

Specifies if a carriage return (CR) should be sent after a forms feed (FF) when VFC=YES is specified. CR has no meaning unless VFC=YES is also specified. If omitted, the default is NO.

EPC=Yes**EPC=No**

Specifies if Early Print Complete should be used by the line driver. If omitted, the default is NO.

FEATure=

Specifies that the printer has the following features:

AT

APL/Text feature

DA

Data Analysis-APL feature

DBCS

Double-Byte Character Set feature

DBCSAT

AT and DBCS features

Lpage=66**Lpage=nn**

Specifies the number of lines per page on the type of form inserted in the 3270 printer. The value may be from 0 to 99. A value of 0 specifies that no page ejects will be done when a page eject is read in the file. If this keyword is omitted, the length of a page defaults to 66.

Note: This value is used by RSCS only to determine the current vertical position on a page while printing a file. Any actual page ejects are determined by Skip-to-Channel-1 CCWs within the file. If there are no Skip-to-Channel-1 CCWs in the file, no page ejects are done, except between files. The actual number of lines contained in the file between consecutive Skip-to-Channel-1 CCWs must be equal to or less than this value.

OVP=Yes**OVP=No**

Specifies if the link will allow overprinting to occur (such as underscored words or overstruck characters for highlighting). If omitted, the default is to allow overprinting. Specify NO for those printers that do not support the 3270 CR (Carriage Return) order.

Ppos=132**Ppos=nnn**

Specifies the maximum number of print positions available on the 3270 printer. This value may be between 1 and 220, depending on the actual printer. Consult the *Component Description* manual for the specific printer to determine this value. If omitted, the number of print positions defaults to 132.

PPOS is a hardware maximum determined by the number of characters the printer *can* print on each line, independent of the size of the form used in the printer. Each printer has its own PPOS value. If the PPOS value you specify does not match the printer's PPOS value, some types of printed output (including separator pages) will not print correctly.

SEP=

Specifies if a separator page will be inserted before each print file and sent across the link.

Yes

For RSCS-style separator pages.

No

For no separator pages. This is the default.

VM

For VM-style separator pages.

Trans=

Specifies the default translation mode for files being transmitted to a printer having the DA, AT, DBCS, or DBCSAT feature, or specifies that unprintable characters should not be translated into blanks (the ASISCC operand). Specifying TRANS tells RSCS that all files may contain the special characters you are indicating and tells RSCS to translate these special characters accordingly. However, the default translation mode you specify on TRANS can be overridden by using the PRT option of the CP TAG command.

APL

If RSCS detects internally represented EBCDIC special APL characters in a file, RSCS will translate those characters into the appropriate two-byte I/O interface codes.

APL2

If RSCS detects internally represented EBCDIC special APL2® character in a file, RSCS will translate those characters into the appropriate two-byte I/O interface codes.

This mode is only valid when FEATURE is set to DBCSAT or AT. If TRANS=APL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

ASISCC

RSCS will not translate unprintable characters into blanks. This lets the user pass SCS orders to the remote printer.

TRANS=ASISCC means no translation occurs as a default. If a file needs translation, then override this using the TAG command (assuming you used the correct setting for the FEATURE operand of the START command.)

If the FEATURE option is not specified, only ASISCC, GRAPH, GRAF or NOTR can be used on the TAG PRT command.

Note: TRANS=ASISCC can be specified without a FEATURE option.

DBCS

RSCS will verify double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

DBCSAPL

RSCS will translate internally represented EBCDIC special APL characters into the appropriate two-byte I/O interface codes and will verify all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

DBCSAPL2

All files containing internally represented EBCDIC special APL2 characters are translated to the appropriate two-byte I/O interface codes. This mode is only valid when FEATURE is set to DBCSAT. If TRANS=DBCSAPL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

DBCSTEXT

RSCS will translate internally represented EBCDIC special TEXT characters into the appropriate two-byte I/O interface codes and will verify double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

Text

RSCS will translate internally represented EBCDIC special TEXT characters into the appropriate two-byte I/O interface codes.

VFC=No**VFC=Yes**

Specifies that the printer has the vertical forms control feature. When YES is specified, the page length must be set manually on the printer and the LPAGE value must match that setting. If omitted, the default is no vertical forms control, and vertical spacing is achieved through multiple New Line orders. YES is valid only for those printers that also support the 3270 CR (Carriage Return) order.

Note: Generally, it is recommend you specify VFC=YES. If users send GDDM or IPDS graphic files to this printer, you *must* specify VFC=YES to ensure that the pages align correctly.

XML=No**XML=Yes**

Specifies that the link is to include an extra NL (New Line) order after each line that is as long as the maximum number of print positions. Specify YES for those printers that do not generate an extra new line function for a maximum length line.

PRTWtm

The time, in seconds (1-600), between recovery attempts due to an unattached device. If not specified, the default is 10 seconds.

PRTRet

The number of hours (1-100) the link will attempt error recovery due to an unattached device before terminating. If not specified, the default is 1 hour.

MSG=Yes**MSG=No**

Specifies if RSCS will allow the TN3270E-type link to dequeue messages destined for transmission to the printer.

The following notes explain the use of the PARM statement for a TN3270E Type link:

1. When you define a form name (using the FORM statement), you can define the following characteristics:
 - Separator page style
 - Line length
 - Page length

If you specify a form name when starting a link, be aware that the form name characteristics always override any SEP, PPOS, and LPAGE specifications.

2. If you specify an incorrect combination of TRANS and FEATURE settings, the SCO receives error message 807E and the link is deactivated.

Table 32. Correct Combinations for TRANS and FEATURE Settings

TRans=	FEATure= DA	FEATure= AT	FEATure= DBCS	FEATure= DBCSAT
APL	X	X		X
APL2		X		X
TEXT	X	X		X
DBCS			X	X
DBCSAPL				X
DBCSAPL2				X
DBCSTEXT				X
ASISCC	X	X	X	X

Note: If APL, APL2, or TEXT is specified for the TRANS operand, and the FEATURE operand is set to DBCSAT, then there is no DBCS translation of the file, only APL, APL2, or TEXT. The DBCS translation will only occur with APL, APL2, or TEXT translation when the TRANS setting is DBCSAPL, DBCSAPL2, or DBCSTEXT.

Note: TRANS=ASISCC does not require the FEATURE operand to be specified.

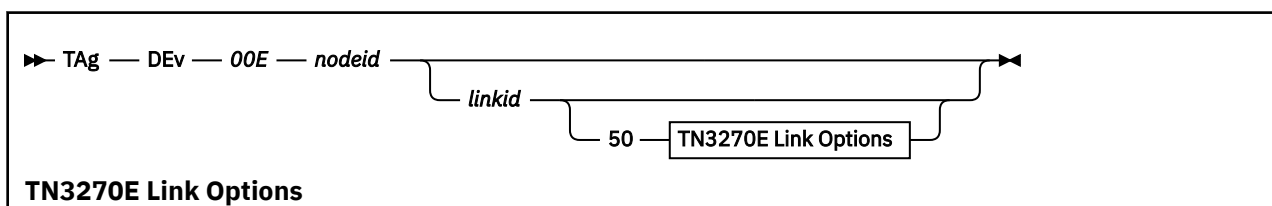
TAG Command for a TN3270E printer

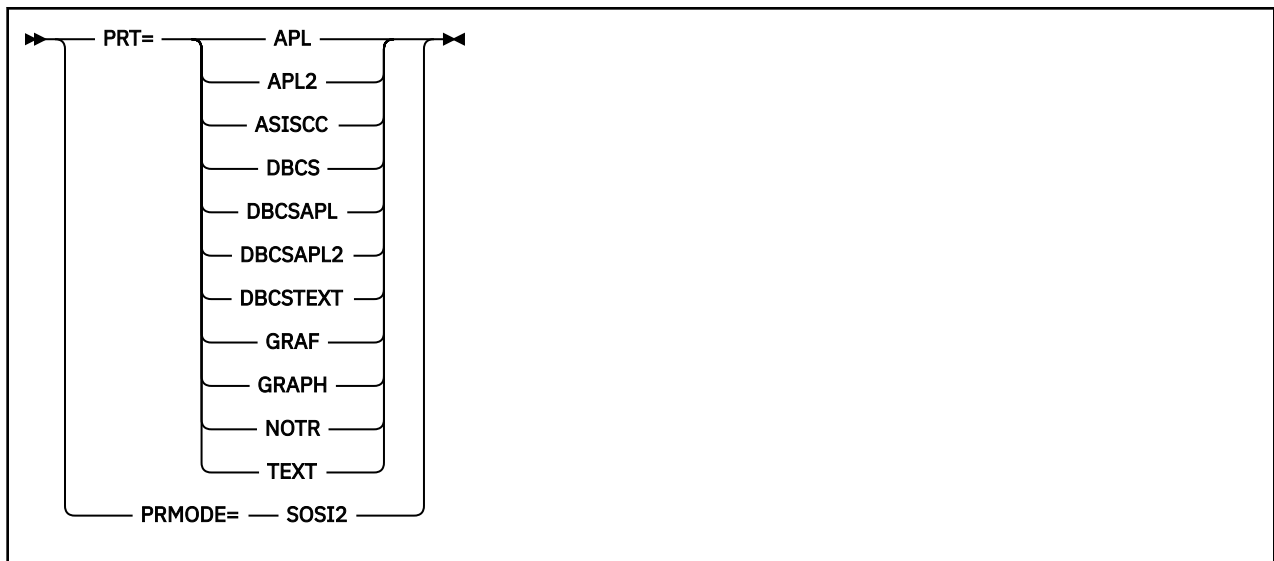
The options for a TAG command to a TN3270E printer are described in the following sections. RSCS processes all the options until it meets a left parenthesis or the end of the command.

Do not include an option more than once in the same TAG command. If you do, RSCS uses the **last** value specified. For example, if you specify the following PRT options on a TAG command to a TN3270E printer, the value of PRT will be DBCSTEXT:

```
tag dev ... prt=asiscc ... prt=dbcstext
```

The following is a syntax diagram of the TAG command:



**nodeid**

Specifies the destination node ID for output created by the virtual device. You must specify **SYSTEM** if you also specify **DEST** on the **SPOOL** command, to allow **DEST** to have effect.

linkid

Identifies the virtual machine link ID, the directly attached workstation, or the directly attached TN3270E printer link ID at the destination node (*nodeid*) that is to receive the output generated by the virtual device.

If the link ID is omitted or specified as **SYSTEM**, the file will be transferred to the system printer or punch. You must specify a link ID (or **SYSTEM**) if you also specify a priority. You must specify **SYSTEM** if you also specify **DEST** on the **SPOOL** command, to allow **DEST** to have effect.

PRT=

Overrides any translation mode (**TRANS**) specification of **APL**, **APL2***, **TEXT**, or **DBCS** on the **RSCS START** command for TN3270E-type links. If no **APL-**, **TEXT-**, or **DBCS-**related **PARM** operands were specified when the link was started, **PRT** is ignored, except for **NOTR**, **GRAF**, **GRAPH**, or **ASISCC**.

If **PRT** is not specified, the **START PARM** or **OPARM** (if specified) **TRANS** setting is the default translation used for the file.

Alternatively, **PRT** can be used to identify a **GDDM** (Graphical Data Display Manager) spool file.

The **PRT** can have the following values

APL

Specifies that internally represented EBCDIC special APL characters will be translated to the appropriate two-byte I/O interface codes.

APL2

All files containing internally represented EBCDIC special APL2 characters are translated to the appropriate two-byte I/O interface codes. This mode is only valid when **FEATURE** is set to **DBCSAT** or **AT**. If **PRT=APL2** is specified with any other **FEATURE** setting, it is marked as an incorrect combination and an error message is created.

ASISCC

RSCS will not translate unprintable characters in files into blanks. This lets the user pass 3270 orders to the remote printer.

DBCS

Specifies that the file contains Double-Byte Character Set (**DBCS**) data.

DBCSAPL

Specifies that all files containing internally represented EBCDIC special APL characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character

strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

DBCSAPL2

Specifies that all files containing internally represented EBCDIC special APL2 characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

This mode is only valid when FEATURE is set to DBCSAT. If PRT=DBCSAPL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

DBCSTEXT

Specifies that all files containing internally represented EBCDIC special text characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

GRAPH

GRAF

Means that the file contains GDDM (Graphical Data Display Manager) output records. For translation to occur the file must be in *punch* format.

NOTR

Means that the internally represented EBCDIC special APL, APL2, or TEXT characters are not to be translated to the two byte I/O interface codes for this file. Only 3270 translation will occur for the removal of 3270 control characters.

TEXT

Specifies that internally represented EBCDIC special text characters will be translated to the appropriate two-byte I/O interface codes.

PRMODE=

Specifies if a blank character should print before and after a DBCS character. PRMODE=SOSI2 for TN3270E will result in the SO and SI being replaced with an SA order for DBCS and a reset and will occupy no space in the output line. When PRMODE is not specified, then the SO, SI will occupy 1 position in an output line.

SOSI2

Does not print a blank before and after a DBCS character string.

Chapter 13. Configuring the SMTP Server

The Simple Mail Transfer Protocol (SMTP) server virtual machine can be configured to:

- operate as either an “open” or a *secure* mail gateway between TCP/IP network users, and users located on an RSCS network that is attached to the local host. This allows, for example, OfficeVision users to exchange mail with UNIX users through the VM TCP/IP SMTP gateway.
- direct mail to a *mailer* server virtual machine, or to a specific remote SMTP server for processing.
- process mail using mail aliases, forwarding, and distribution lists defined in an SMTP NAMES file.
- change mail header addresses and specify header address transformations.

To configure the SMTP server, you must perform the following steps:

SMTP Server Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the system (CP) directory for the SMTP server. 3. Update the DTCPARMS file for the SMTP server. 4. Update the TCPIP DATA file for domain name resolution. 5. Customize the SMTP CONFIG file. 6. Perform advanced SMTP server configuration, if needed.

Dynamic System Operation: The SMTP server provides a VM Special Message (MSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information, see [“Dynamic Server Operation: MSG Interface to the SMTP Server”](#) on page 420.

Step 1: Update PROFILE TCPIP

Include the SMTP server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The SMTP server is then automatically started when TCPIP is initialized. The IBM default user ID for this server is SMTP. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
SMTP    0
```

The SMTP server requires that port TCP 25 be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
25 TCP SMTP ; SMTP Server
```

Step 2: Update the System (CP) Directory for the SMTP Server

The SMTP virtual machine must be authorized to use the *SPL system service for reading spool files. To use this service, the following CP directory statement must be added for each SMTP server:

```
IUCV *SPL
```

Step 3: Update the DTCPARMS File

When the SMTP server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the SMTP server are:

```
:nick.SMTP
:Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

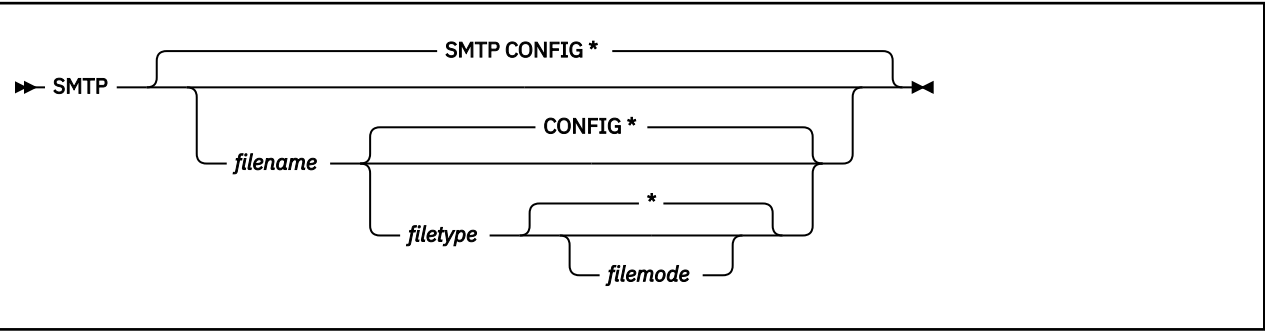
For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Note: You should modify the DTCPARMS file for the SMTP server if you use a configuration file other than SMTP CONFIG.

SMTP Command Operands (:Parms. Parameters)

SMTP services are initiated using the SMTP command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with an SMTP server definition. For more information about this command and its operands, see [“SMTP Command Syntax”](#) on page 376.

SMTP Command Syntax



Operands

- filename**
The file name of the SMTP server configuration file. The default file name is SMTP.
- filetype**
The file type of the configuration file. The default file type is CONFIG.
- filemode**
The file mode of the configuration file. The default file mode is *.

Step 4: Update the TCPIP DATA File for Domain Name Resolution

Specify the method used by the SMTP server to resolve host names (either domain name server or local site tables, or both), and the order in which these methods are used, by using the DOMAINLOOKUP statement in the TCPIP DATA configuration file. If local site tables are used for name resolution, SMTP uses the ETC HOSTS file, if present, or the site tables created by the HOSTS LOCAL file, if ETC HOSTS is not present. For more information, see [“DOMAINLOOKUP statement”](#) on page 14.

Note: SMTP host name resolution uses the domain specified on the DOMAINORIGIN statement; SMTP does not use the domains specified on the DOMAINSEARCH statement.

If domain name server is used (DNS or DNSONLY is specified on the DOMAINLOOKUP statement), then define the name servers using the NSINTERADDR statements in the TCPIP DATA. For more information, see [Chapter 3, “Defining the TCP/IP System Parameters,”](#) on page 13.

To obtain detailed information about how the SMTP server resolves a particular host name (for debugging purposes), you can use the SMTP **TRACE RESOLVER** configuration file statement or SMSG command;

these are described in more detail in [“TRACE Statement”](#) on page 402 and [“Privileged User SMSG TRACE Command”](#) on page 435.

When SMTP uses a name server, it asks for the MX records for the host to which it is trying to connect. If such MX records are unavailable, the A records for the host are used. If such MX records are available, the AAAA and A records for the host are used. For more information about MX records, see [“Use of MX Records”](#) on page 377 or *RFC 974, Mail Routing and the Domain System*.

Step 5: Customize the SMTP CONFIG File

The SMTP configuration file, SMTP CONFIG, defines how the SMTP server is to operate, and what services it provides. See [“SMTP Server Configuration File Statements”](#) on page 378 for detailed information about how to specify entries within this file. A sample SMTP configuration file is provided as SMTP SCONFIG on the TCPMAINT 591 disk. Your customized SMTP configuration file should be copied to the TCPMAINT 198 minidisk as SMTP CONFIG. [Table 33 on page 378](#) provides a summary of the configuration statements.

Step 6: Additional SMTP Server Considerations

Before you configure the SMTP server using the statements described in [“Step 5: Customize the SMTP CONFIG File”](#) on page 377, you may want to review the information in the following sections:

- [“SMTP Server Exits”](#) on page 407.
- [“Configuring a TCP/IP-to-RSCS Mail Gateway”](#) on page 407
- [“Configuring a TCP/IP-to-RSCS Secure Mail Gateway”](#) on page 409
- [“Defining Nicknames and Mailing Lists Using the SMTP NAMES File”](#) on page 411
- [“Customizing SMTP Mail Headers”](#) on page 412.

These sections discuss additional configuration steps and files that are necessary for the SMTP server to run correctly when using these features.

The sections that follow provide information about how MX records defined within the Domain Name System and the classification of a mail recipient can affect SMTP server operations.

Use of MX Records

MX records are a type of resource record used in the Domain Name System. MX records direct the SMTP server to deliver mail to alternative hosts. They are obtained from a domain name server. If SMTP is configured to not use a name server, MX records are not used.

For example, if SMTP wants to send mail to USER@HOST, it checks the name server for MX records and finds the following:

HOST	MX	0	HOST
HOST	MX	5	HOST - BACKUP1
HOST	MX	10	HOST - BACKUP2

SMTP delivers the mail to the record (host) that has the lowest count; in this example, mail is delivered directly to HOST. If HOST cannot receive the mail, SMTP tries to deliver it to HOST - BACKUP1. If HOST - BACKUP1 cannot receive the mail, it then tries to deliver it to HOST - BACKUP2. If none of these hosts can receive the mail, SMTP stores the mail and queues it for later delivery, at which time this delivery process is repeated.

If SMTP does not find MX records for a host, it delivers mail to only the primary host.

Local versus Non-local Mail Recipients

When SMTP processes a piece of mail, it determines whether the recipient of that mail is a *local* or a *non-local* user, to allow proper handling of that mail. For example, the result of this determination can affect the mail forwarding processing performed by SMTP.

When a recipient is determined to be a *local* user, the z/VM SMTP server then handles the delivery of that mail itself, possibly through the use of a local mailer, or other network services. SMTP considers a user to be a local user if that user is either:

- a user ID on the local z/VM system
- defined in its SMTP NAMES file, through either a mail alias, a mail forwarding entry, or a mail distribution list
- a user located on an RSCS network that is attached to the local host (applies only to SMTP mail gateway configurations)

If these criteria do not apply to the mail recipient, SMTP considers that user to be a *non-local* user, and will take necessary actions to process that user's mail.

SMTP Server Configuration File Statements

Table 33 on page 378 provides a summary of the configuration file statements. Keep in mind the following when configuration statements are specified:

- Tokens are delimited by blanks and record boundaries.
- All characters to the right of, and including, a semicolon are treated as comments.

Table 33. SMTP CONFIG Configuration Statements

Statement	Description	Location
ALTRSCSDOMAIN	Specifies an alternative domain name of the RSCS network, if SMTP is running as a mail gateway.	“ALTRSCSDOMAIN Statement” on page 381
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.	“ALTTCPHOSTNAME Statement” on page 381
BADSPoolFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.	“BADSPoolFILEID Statement” on page 381
DBCS	Specifies that SMTP should perform DBCS code conversion on mail.	“DBCS Statement” on page 381
FILESPerCONN	Specifies the number of files (notes) at which another connection will be opened to a remote host.	“FILESPerCONN Statement” on page 383
FINISHOPEN	Specifies the SMTP wait time for connection.	“FINISHOPEN Statement” on page 383
FORWARDMAIL	Specifies whether or not to forward mail to non-local users, or identifies a user exit to control mail forwarding.	“FORWARDMAIL Statement” on page 383
GATEWAY	Specifies operation of SMTP as a gateway.	“GATEWAY Statement” on page 384
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.	“INACTIVE Statement” on page 385
IPMAILERADDRESS	Specifies the address on which SMTP queues mail when it cannot resolve the recipient's address.	“IPMAILERADDRESS Statement” on page 385
LOCALFORMAT	Specifies the spool file format for local host mail delivery.	“LOCALFORMAT Statement” on page 386
LOG	Directs SMTP to log all SMTP traffic.	“LOG Statement” on page 387

Table 33. SMTP CONFIG Configuration Statements (continued)

Statement	Description	Location
MAILER	Identifies a virtual machine to receive mail for various classes of mail recipients.	“MAILER Statement” on page 387
MAILHOPCOUNT	Specifies a limiting value that is used by SMTP to detect a mail delivery loop condition and then cease delivery attempts for a mail item.	“MAILHOPCOUNT Statement” on page 388
MAXCONNPERSITE	Specifies the maximum number of connections that can be opened to a remote host concurrently.	“MAXCONNPERSITE Statement” on page 389
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over TCP/IP connections.	“MAXMAILBYTES Statement” on page 389
NOLOG	Turns off logging of mail transactions.	“NOLOG Statement” on page 389
ONDISKFULL	Specifies a set of CP commands for SMTP to execute when specified SMTP 191 disk-full thresholds are crossed.	“ONDISKFULL Statement” on page 390
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP/IP connections over which SMTP actively delivers mail.	“OUTBOUNDOPENLIMIT Statement” on page 391
PORT	Specifies an alternative port number for SMTP server during testing.	“PORT Statement” on page 391
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.	“POSTMASTER Statement” on page 391
RCPTRESPONSEDELAY	Specifies the amount of time the SMTP server delays responding to the RCPT commands.	“RCPTRESPONSEDELAY Statement” on page 392
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.	“RESOLVERRETRYINT Statement” on page 393
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.	“RESTRICT Statement” on page 393
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.	“RETRYAGE Statement” on page 394
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP/IP host.	“RETRYINT Statement” on page 394
REWRITE822HEADER	Specifies whether SMTP should rewrite the RFC 822 headers of mail arriving from the RSCS side of the mail gateway.	“REWRITE822HEADER Statement” on page 394
RSCSDOMAIN	Specifies the domain name of the RSCS network (if SMTP is running as a mail gateway).	“RSCSDOMAIN Statement” on page 395
RSCSFORMAT	Specifies the spool file format for mail delivered to recipients on the RSCS network.	“RSCSFORMAT Statement” on page 395

Table 33. SMTP CONFIG Configuration Statements (continued)

Statement	Description	Location
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP/IP network sites and RSCS network sites.	“SECURE Statement” on page 396
MSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP MSG commands.	“MSGAUTHLIST Statement” on page 396
SMTPCMDS	Defines the characteristics of the SMTP command user exit and indicates whether or not the exit is to be enabled (on) or disabled (off) when the server completes initialization.	“SMTPCMDS Statement” on page 397
SOURCEROUTES	Specifies whether or not SMTP will honor source routes.	“SOURCEROUTES Statement” on page 399
SUPPRESSNOTIFICATION	Specifies whether the SMTP server should suppress messages that are otherwise sent back to the sender when batch SMTP mail is received at the server and/or when batch SMTP mail is delivered to a recipient.	“SUPPRESSNOTIFICATION Statement” on page 400
TEMPERORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.	“TEMPERORRETRIES Statement” on page 400
TLS	Specifies the TLS security level for the SMTP server.	“TLS Statement” on page 401
TLSLABEL	Specifies the TLS label to be used by SMTP when securing connections using TLS.	“TLSLABEL Statement” on page 402
TRACE	Specifies which type of tracing should be enabled during server initialization.	“TRACE Statement” on page 402
VERIFYBATCHSMTPSENDER	Specifies that verification (using spool file TAG information) should be performed on batch SMTP sender (MAIL FROM:) information.	“VERIFYBATCHSMTPSENDER Statement” on page 403
VERIFYCLIENT	Specifies whether or not client verification is to be performed. Client verification can be performed using the built-in client verification function or through a user exit.	“VERIFYCLIENT Statement” on page 404
VERIFYCLIENTDELAY	Specifies the amount of time (in seconds) that SMTP will wait for the domain name system to respond during client verification.	“VERIFYCLIENTDELAY Statement” on page 405
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender.	“WARNINGAGE Statement” on page 405
8BITMIME	Specifies the file name of the translation table to be used for 8-bit MIME support.	“8BITMIME Statement” on page 406

ALTRSCSDOMAIN Statement

The ALTRSCSDOMAIN statement specifies an alternate domain name for an RSCS network (if SMTP is configured to operate as a mail gateway). This alternate domain name is used in the same manner, but in addition to any domain name defined using the RSCSDOMAIN statement. Only one ALTRSCSDOMAIN statement can be specified within the SMTP configuration file.

```
➤➤ ALTRSCSDOMAIN — domain ➤➤
```

Operands

domain

The alternate domain name of the RSCS network. The alternative RSCS domain name is a string of 1 to 64 alphanumeric characters.

ALTTCPHOSTNAME Statement

The ALTTCPHOSTNAME statement specifies an alternate fully qualified host name by which SMTP knows the local host. Mail sent to users at *hostname* are treated as if they were local users.

```
➤➤ ALTTCPHOSTNAME — hostname ➤➤
```

Operands

hostname

The alternate TCP/IP host name of this host.

BADSPoolFILEID Statement

The BADSPoolFILEID statement specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail. You can specify the BADSPoolFILEID statement only once.

```
➤➤ { BADSPoolFILEID TCPMAINT }
   { BADSPoolFILEID user_id } ➤➤
```

Operands

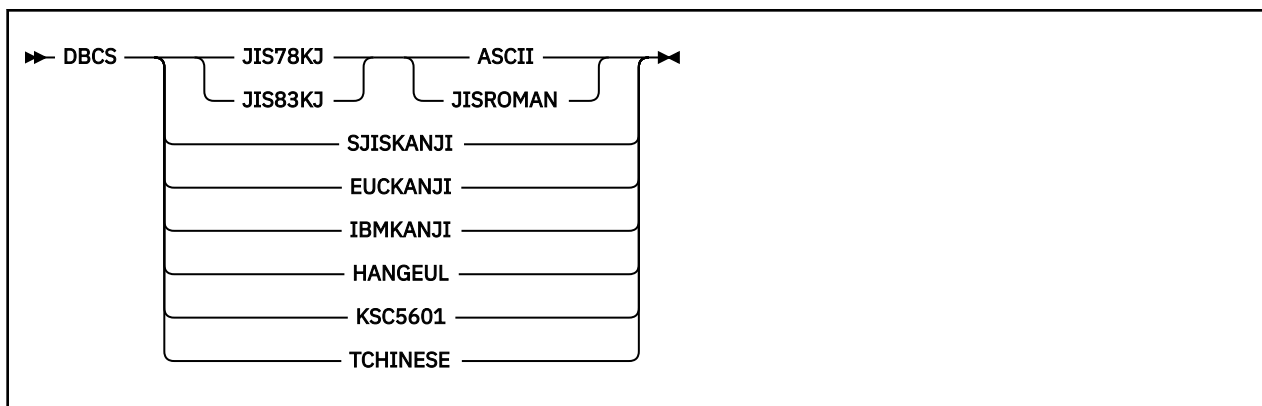
user_id

The user ID on the local system to which bad spool files and looping mail are delivered; *user_id* should be a maximum of 8 characters. The default is TCPMAINT.

DBCS Statement

The DBCS statement specifies that SMTP is to perform DBCS code conversion on processed mail. The parameters of the DBCS statement determine which translation table should be used in the conversion.

The following example shows the format of the DBCS statement.



Operands

JIS78KJ

Indicates IBM Kanji to JIS 1978 Kanji conversion is to be performed. The JIS 1978 Shift-Out codes are ESC, \$, and @. SMTP will load the JIS78KJ DBCS translation table from the TCPKJBIN binary translate table file.

JIS83KJ

Indicates IBM Kanji to JIS 1983 Kanji conversion is to be performed. The JIS 1983 Shift-Out codes are ESC, \$, and B. SMTP will load the JIS83KJ DBCS translation table from the TCPKJBIN binary translate table file.

ASCII

Indicates mail is to be shifted in ASCII code from JIS Kanji code. The ASCII Shift-In codes are ESC, (, and B.

JISROMAN

Indicates mail is to be shifted in JISRoman code from JIS Kanji code. The JISRoman Shift-In codes are ESC, (, and J.

SJISKANJI

Indicates IBM Kanji to Shift-JIS Kanji conversion is to be performed. SMTP will load the SJISKANJI DBCS translation table from the TCPKJBIN binary translate table file.

EUCKANJI

Indicates IBM Kanji to Extended UNIX Code Kanji conversion is to be performed. SMTP will load the EUCKANJI DBCS translation table from the TCPKJBIN binary translate table file.

IBMKANJI

Indicates IBM (EBCDIC) Kanji conversion is to be used. This option causes **no** conversion to be performed on the body of the mail; it may be used for sending and receiving mail in EBCDIC. If this option is selected, other SMTP servers on the same network should all be configured to perform IBMKANJI conversions. If IBMKANJI is specified, and LOCALFORMAT or RSCSFORMAT is set to PUNCH, then mail received in ASCII may be folded to inconsistent record lengths. LOCALFORMAT and RSCSFORMAT should be set to NETDATA in this case.

Indicates IBMKANJI transfer type does not require any translate table to be loaded.

HANGEUL

Indicates IBM Hangeul to Hangeul PC code conversion is to be performed. SMTP will load the HANGEUL DBCS translation table from the TCPHGBIN binary translate table file.

KSC5601

Indicates IBM Hangeul to KSC-5601 PC code conversion is to be performed. SMTP will load the KSC5601 DBCS translation table from the TCPHGBIN binary translate table file.

TCHINESE

Indicates IBM Traditional Chinese to Traditional Chinese 5550 PC code conversion is to be performed. SMTP will load the TCHINESE (5550) DBCS translation table from the TCPCHBIN binary translate table file.

The SBCS translation table, STANDARD TCPXLBIN, will be used to convert mail headers, with the body of the mail being converted using a DBCS translation table from SMTP TCPKJBIN, SMTP TCPHGBIN, or SMTP TCPCHBIN. If SMTP TCPKJBIN, SMTP TCPHGBIN, or SMTP TCPCHBIN, cannot be found, then SMTP will use STANDARD TCPKJBIN, STANDARD TCPHGBIN, or STANDARD TCPCHBIN.

DBCS conversion is performed only on outgoing and incoming mail to and from TCP/IP-connected hosts. Mail spooled to SMTP for the local host is delivered directly, without any DBCS code conversion.

Note: For more information on loading and customizing DBCS translate tables, see [“Customizing DBCS Translation Tables”](#) on page 673.

FILESPERCONN Statement

The FILESPERCONN statement specifies the number of files (notes) backed up to a remote host at which another connection will be opened to help deliver mail. A new connection will be opened every FILESPERCONN files backed up to a remote host up to the maximum value specified in the MAXCONNPERSITE configuration file statement.



Operands

files

An integer in the range of 1 through 1,000 indicating the number of files at which another connection will be opened to a remote host. The default is 5 files.

FINISHOPEN Statement

The FINISHOPEN statement specifies the number of seconds SMTP waits while trying to establish a connection to a foreign site. After the specified number of seconds, SMTP aborts the connection.



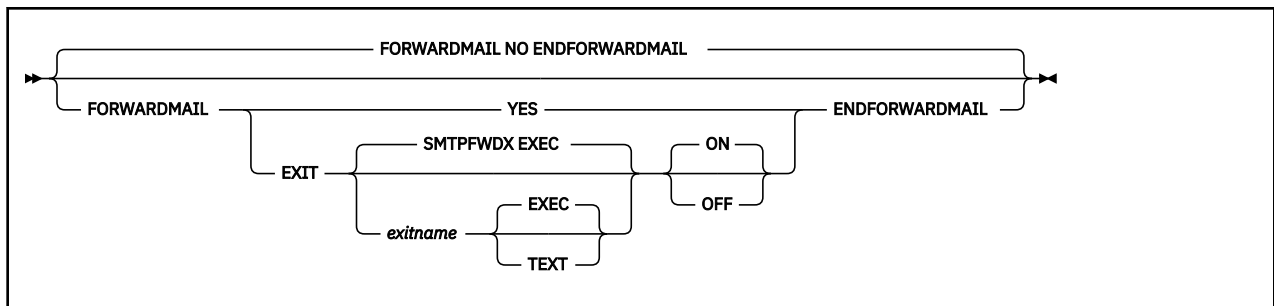
Operands

seconds

An integer in the range of 1 through 86,400 (24 hours) indicating the number of seconds to wait for a connection to open. The default FINISHOPEN time-out is 120 seconds.

FORWARDMAIL Statement

The FORWARDMAIL statement is used to indicate whether or not to forward mail to non-local users, or to identify a user exit to be used to control mail forwarding. For information on using the mail forwarding exit, see the [z/VM: TCP/IP Programmer's Reference](#).



Operands

NO

Indicates that no mail forwarding is to be performed. When SMTP determines that the recipient is not on the local system, the RCPT TO: command will be rejected; this is the default.

YES

Indicates mail forwarding is to be performed.

EXIT

Indicates a mail forwarding exit routine is being defined.

exitname

The name of the exit routine associated with this command; the default exit name is SMTPFWDX.

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT

Indicates the exit routine name specified on this command is the name of a text deck.

ON

Indicates the specified exit (being defined with this command) is to be enabled (turned on).

OFF

Indicates the specified exit (being defined with this command) is to be disabled (turned off).

Examples

- The following SMTP configuration file entry will enable the mail forwarding exit routine SMTPFWDX EXEC.

```
ForwardMail
  exit smtpfwdx exec on
EndForwardMail
```

When this entry is processed, the following text is displayed during server initialization:

```
Forward Mail                : Exit SMTPFWDX EXEC ON
```

- This next entry defines the mail forwarding exit routine SMTPFWDX TEXT, but disables its use once SMTP server initialization is complete. Thus, mail forwarding will be allowed and performed.

```
ForwardMail
  exit smtpfwdx text off
EndForwardMail
```

When this entry is processed, the following text is displayed during server initialization:

```
Forward Mail                : Yes (Exit SMTPFWDX TEXT OFF)
```

GATEWAY Statement

The GATEWAY statement specifies that SMTP is to operate as a mail gateway between TCP/IP network sites and RSCS network sites (provided the host system is connected to both a TCP/IP network and an RSCS network).

If GATEWAY is specified, SMTP accepts mail addressed to users on RSCS hosts defined in the SMTPRSCS HOSTINFO file. For more information about setting up a gateway node, see [“Configuring a TCP/IP-to-RSCS Mail Gateway”](#) on page 407. If GATEWAY is not specified, SMTP rejects all mail that arrives from RSCS.

➤ GATEWAY ➤

Operands

The GATEWAY statement has no operands.

INACTIVE Statement

The INACTIVE statement specifies the period of inactivity (in seconds) after which SMTP considers a connection to be broken (that is, unusable). After the specified amount of inactive time, SMTP closes the connection.

➤ INACTIVE 180 ➤
 ➤ INACTIVE *seconds* ➤

Operands

seconds

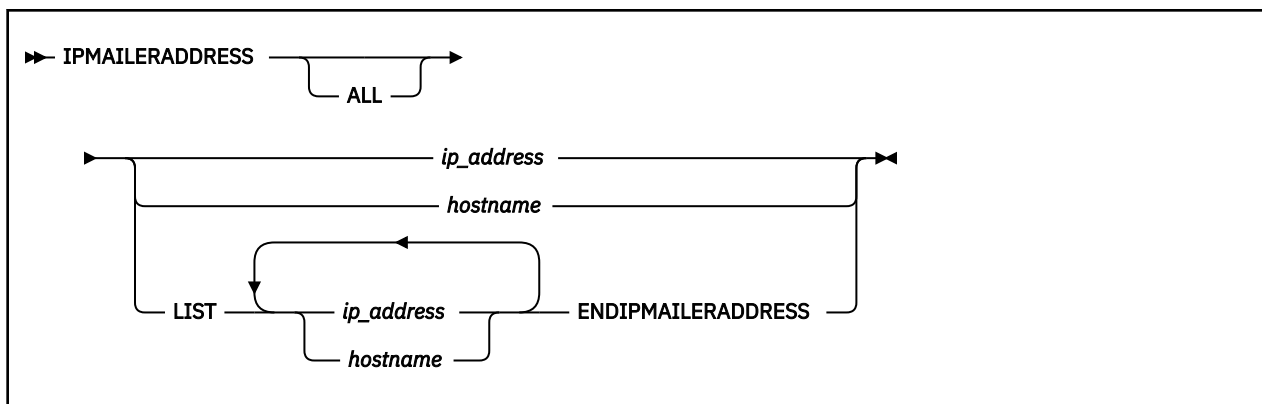
An integer, in the range of 1 through 86,400 (24 hours), that specifies the number of seconds after which SMTP considers a connection to be broken. The default INACTIVE time-out period is 180 seconds.

IPMAILERADDRESS Statement

The IPMAILERADDRESS statement specifies an IP address to which SMTP queues mail when it cannot resolve a mail recipient address.

Note: The IPMAILERADDRESS statement and the UNKNOWN parameter of the MAILER statement are mutually exclusive. The SMTP server will not initialize if both are specified in the SMTP configuration file.

IPMAILERADDRESS is useful when your VM system does not have full connectivity to or knowledge of other networks. It allows you to direct mail to a different SMTP server that may be able to deliver mail to requested hosts that reside on these other networks.



Operands

ip_address

The IPv4 or IPv6 internet address of the host to which SMTP is to direct mail when it cannot resolve a mail recipient address. If the ALL operand is specified, all non-local mail will be directed to this host.

hostname

The hostname to which SMTP is to direct mail when it cannot resolve a mail recipient address. If the ALL operand is specified, all non-local mail will be directed to this host.

LIST

Indicates that a list of IP addresses or hostnames will be given to which SMTP will direct a piece of mail when it cannot resolve a mail recipient address. SMTP will attempt to direct the mail to the first IP address or hostname in the list. If unable to connect to that host, SMTP will proceed through the list sequentially until the mail is redirected or the list is exhausted.

ALL

Indicates that SMTP should direct all non-local mail to the specified IP address or hostname. When this option is configured, the SMTP server will not attempt to resolve recipient host names on any mail that is being processed by the server. If the recipient host name is anything other than the local host, the mail will be sent to the host specified on the IPMAILERADDRESS statement (whether the mail was generated on the local host and is going out, or the mail came into the server from the outside and is being forwarded).

Usage Notes

1. If an IPMAILERADDRESS statement is coded, it must contain at least one valid IP address or hostname. If it does not, then an error message is displayed and the SMTP server is shut down.
2. A loopback address is not considered a valid address when used on the IPMAILERADDRESS statement. This includes any address in the IPv4 loopback range (127.x.x.x), the IPv6 loopback address (::1), and the special loopback hostname ("LOOPBACK").
3. An IPv6 linklocal address is not considered a valid address when used on the IPMAILERADDRESS statement.
4. An IPv6 prefix address is not considered a valid address when used on the IPMAILERADDRESS statement.

LOCALFORMAT Statement

The LOCALFORMAT statement specifies the spool file format for mail delivered to recipients on the local host. The default format is NETDATA.

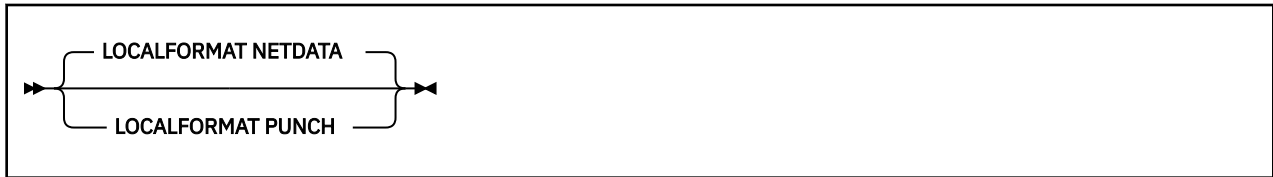
For PUNCH format:

- The spool file is in CMS PUNCH NOHEADER format.

- Records are folded to 80 or fewer characters in length.
- Spool file class M is used.
- The file name is the first 8 characters of the user ID of the sender.
- The file type is MAIL.

For NETDATA format:

- The spool file is in NETDATA format.
- Records can be longer than 80 characters.
- Spool file class A is used.
- The file name is the first 8 characters of the user ID of the sender.
- The file type is NOTE.



Operands

PUNCH

Indicates that records are folded to 80 or fewer characters in length.

NETDATA

Indicates that records can be longer than 80 characters and arrive as MESSAGE-type records. The default format is NETDATA.

LOG Statement

The LOG statement specifies that SMTP is to log information about all SMTP traffic. The origin, sender, and recipients of each piece of mail are written to the log as mail is received and delivered. To turn off logging, use the NOLOG statement.



Operands

DISK

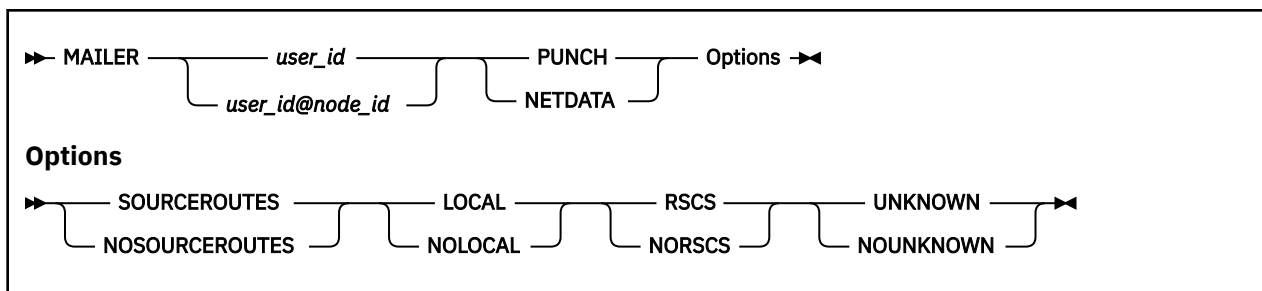
Indicates that log information is to be appended to the SMTP log file; DISK is the default.

SPOOL

Indicates that log information is to be written to the console.

MAILER Statement

The MAILER statement identifies a virtual machine to receive mail for various classes of mail recipients. There are three possible formats that the MAILER virtual machine will accept. If PUNCH or NETDATA format is specified, then the MAILER virtual machine must exist on the local node or a node on the RSCS network and it must accept batch SMTP format spool files.



Operands

user_id

The user ID of a MAILER virtual machine; this machine is presumed to reside on the local RSCS node.

user_id@node_id

The user ID and RSCS node of a MAILER virtual machine.

PUNCH

Indicates that the remote MAILER virtual machine can accept only PUNCH format spool files. BSMTF header records longer than 80 characters are split, and an EBCDIC newline character (X'15') is placed in column 80 to indicate that the record is continued. Records within the body of the mail longer than 80 characters are split across multiple PUNCH records.

NETDATA

Indicates that the MAILER virtual machine accepts NETDATA format spool files. The NETDATA protocol automatically handles records longer than 80 characters.

SOURCEROUTES

Indicates that the MAILER virtual machine accepts BSMTF header addresses with source routes.

NOSOURCEROUTES

Indicates that the MAILER virtual machine does not accept source routes in the BSMTF header addresses.

LOCAL

Indicates that mail for local recipients is spooled to the MAILER virtual machine.

NOLOCAL

Indicates that mail for local recipients is spooled directly to the recipients.

RSCS

Indicates that mail for recipients on the RSCS network is spooled to the MAILER virtual machine.

NORSCS

Indicates that mail for recipients on the RSCS network is spooled directly to the recipients.

UNKNOWN

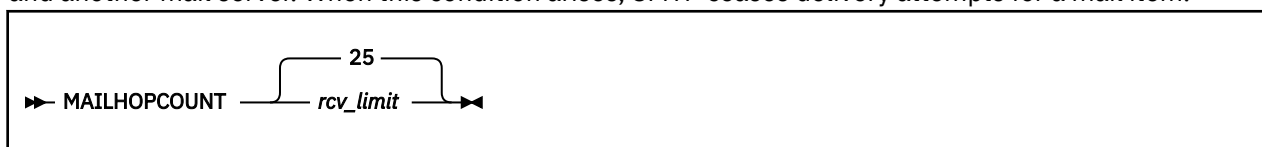
Indicates that mail for recipients on an unknown host is spooled to the MAILER virtual machine.

NOUNKNOWN

Indicates that mail for recipients on unknown hosts is returned to the sender as undeliverable.

MAILHOPCOUNT Statement

The MAILHOPCOUNT statement is used by SMTP to detect a mail delivery loop condition between itself and another mail server. When this condition arises, SMTP ceases delivery attempts for a mail item.



Operands

rcv_limit

An integer in the range of 5 to 300 that specifies the maximum number of Received lines that can exist within the header section of a given mail item. The default is 25.

When the Received lines present in the header section of a given mail item matches the specified *rcv_limit* value, SMTP considers the delivery process for that piece of mail to be in a loop and SMTP stops its attempts to deliver the mail item. The looping mail NOTE6 and ADDR6 files are renamed to have file types of LOOPMAIL and LOOPADDR respectively and sent to the user ID configured in the BADSPOOLFILEID configuration statement. The mail is then disposed of in the SMTP server.

MAXCONNPERSITE Statement

The MAXCONNPERSITE statement specifies the maximum number of connections that can be opened concurrently to one remote host.



Operands

conns

An integer in the range of 1 through 256 indicating the maximum number of concurrent connections that can be opened to a remote host. The default is 10 connections.

MAXMAILBYTES Statement

The MAXMAILBYTES statement specifies the maximum size, in bytes, of mail that is accepted over a TCP connection.



Operands

bytes

The maximum number of bytes to accept. Mail larger than this size that arrives over a TCP connection is rejected. The limits for this statement are 1 and 2,147,483,647. The default byte size is 524,288.

NOLOG Statement

The NOLOG statement turns off the logging of information about SMTP traffic. For more information, see “LOG Statement” on [page 387](#).

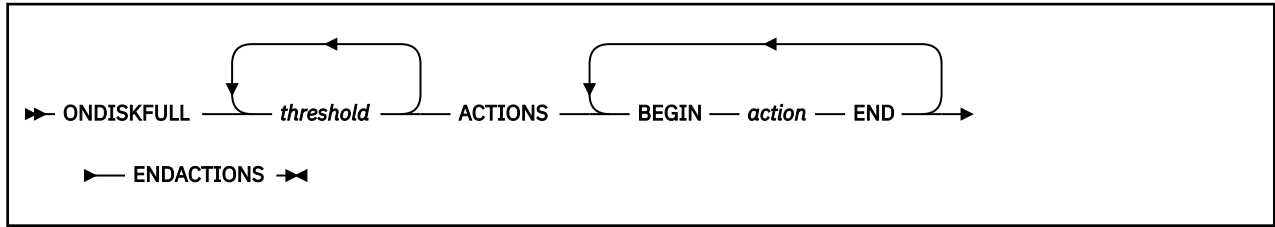


Operands

The NOLOG statement has no operands.

ONDISKFULL Statement

The ONDISKFULL statement specifies a set of CP commands (such as CP MSG or CP SMSG) for SMTP to execute when specified SMTP 191 disk-full thresholds are crossed.



Operands

threshold

The disk-full percentages for which actions should be performed. Positive numbers cause the actions to execute when the SMTP 191 minidisk percentage exceeds the percent specified. Negative numbers cause the actions to execute when the minidisk utilization drops below the percent specified.

action

A CP command that is to be issued when a threshold condition is crossed. To simplify the issuing of informational messages, the special keyword **MESSAGE** can be included as part of the action string. This will cause one of the following text strings (as warranted) to be substituted in place of **MESSAGE**:

```
date time SMTPserver_id at SMTPnode_id - Disk Above nn Percent Full
```

or

```
date time SMTPserver_id at SMTPnode_id - Disk Below nn Percent Full
```

There is no limit on the number of BEGIN *action* END statements that you can specify, but the total length of each action must be less than 255 characters. Note that the special keyword **MESSAGE** accounts for 80 characters. All commands are converted to uppercase and extra blanks are removed. The CP command output for all action commands is suppressed, although nonzero command return codes will be reported. The ENDACTIONS statement ends the ONDISKFULL statement.

In the following example, the ONDISKFULL statement causes messages to be sent to two users (MATT at ENDICOTT and TCPMAINT on the local system) when the SMTP 191 minidisk exceeds 40, 50, 60, 70, 80, and 90 percent full. Messages are also sent when the disk full percentage decreases below these same thresholds.

```

;
; On Disk Full Actions
;
ONDISKFULL 40 50 60 70 80 90 -90 -80 -70 -60 -50 -40 ACTIONS
BEGIN CP SMSG RSCS MSG ENDICOTT MATT *MESSAGE* END
BEGIN CP MSG TCPMAINT *MESSAGE* END
END ACTIONS
;

```

For the above statements, the message that follows would be received by user MATT at ENDICOTT when the 191 minidisk of the SMTP server at WATSON exceeds 50 percent full:

```
From WATSON(SMTP): 03/17/99 21:54 SMTP at WATSON - Disk Above 50 Percent Full
```

whereas the TCPMAINT user (on the local WATSON system) would receive this message:

```
03/17/99 21:54 SMTP at WATSON - Disk Above 50 Percent Full
```

OUTBOUNDOPENLIMIT Statement

The OUTBOUNDOPENLIMIT statement specifies the limit on the maximum number of simultaneous TCP connections over which SMTP can actively deliver mail. By default, SMTP operates with no limits, and opens as many TCP connections as necessary to ensure the fastest possible delivery of mail. The OUTBOUNDOPENLIMIT statement should be specified only if there are limited TCP resources on the system and SMTP is using an unfair portion of these resources.

➤ OUTBOUNDOPENLIMIT — *connections* ➤

Operands

connections

A number in the range of 1 to the maximum number of TCP connections available on your system. If *connections* is out of range, no limit is imposed.

PORT Statement

The PORT statement causes SMTP to listen on a specified TCP connection port. By convention, port number 25 is usually reserved (in the TCPIP server configuration file) for the SMTP server to accept incoming mail requests.

➤ { PORT 25
PORT *port_number* } ➤

Operands

port_number

An integer in the range of 1 through 65,535 that specifies the port number to which SMTP listens. The default is port 25.

POSTMASTER Statement

The POSTMASTER statement identifies the user that should receive mail sent to the “postmaster” of the system where the SMTP server runs. A postmaster is generally responsible for managing the mail system for a site. As such, other users in the internet community might send mail to the postmaster, to report mail problems they believe are associated with that site, or to have mail routed to a recipient (either local or in a different domain) for which the correct address is not known.

➤ { POSTMASTER TCPMAINT
POSTMASTER *user_id*
 user_id@hostname } ➤

Operands

user_id

A user ID on the local system to which mail addressed to “postmaster” should be delivered; the default is TCPMAINT.

user_id@hostname

A user ID and host name to which mail addressed to “postmaster” should be delivered. The supplied *user_id* and *hostname* values may identify either a user ID on a specific RSCS node, or an internet user and host name.

Usage Notes

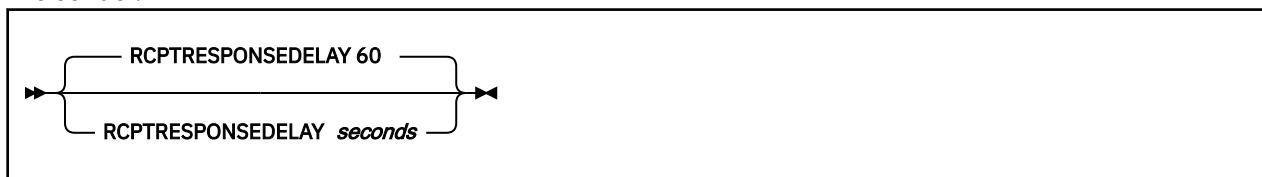
1. If a POSTMASTER statement is not specified, the default value, TCPMAINT, will be used.
2. If SMTP is *not* operating as a secure mail gateway between TCP/IP and RSCS network sites, multiple POSTMASTER statements can be specified. When this is done, code a separate POSTMASTER statement for each user that should receive mail addressed to “postmaster”. No limit is imposed on the number of POSTMASTER statements that can be coded.
3. If SMTP is operating as a secure mail gateway between TCP/IP and RSCS network sites, only one POSTMASTER statement can be specified. Additionally, *user_id* must be a user ID defined for the local system.
4. Since the POSTMASTER commonly forwards mail on behalf of other users, if SMTP is configured to verify batch SMTP sender (MAIL FROM:) information using the VERIFYBATCHSMTPSENDER statement, local POSTMASTER users (specified as a z/VM user ID) and remote POSTMASTER users (specified as a z/VM user ID and RSCS node ID) will automatically be added to the VERIFYBATCHSMTPSENDER exception list.

RCPTRESPONSEDELAY Statement

The RCPTRESPONSEDELAY statement specifies the amount of time the SMTP server delays its response to a RCPT TO: command (supplied by a remote SMTP “sender” host) while host domain resolution of a mail recipient is being performed. If SMTP does not receive a domain name server (DNS) response within the specified period, it *assumes* the host resolution is successful and does the following:

- Sends a 250 OK response to the “sender” SMTP host
- Queues the mail locally, so that asynchronous resolution of the mail recipient can be performed.

If SMTP later determines that the recipient address cannot be resolved, the queued mail is returned to the sender.



Note: When the client sends multiple commands (including the RCPT command) without waiting for the response from the SMTP server, the SMTP server responds immediately to the RCPT command with the 250 OK message. The RCPTRESPONSEDELAY value will be ignored.

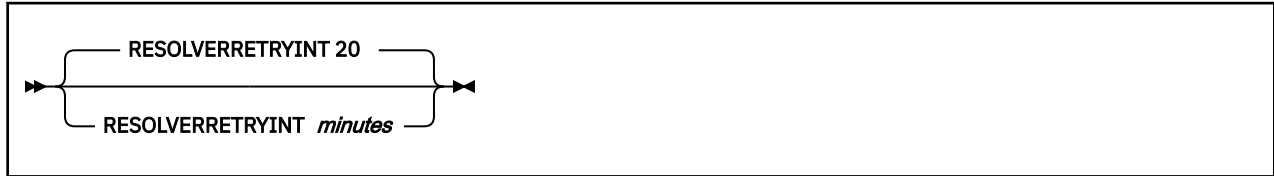
Operands

seconds

The number of seconds SMTP waits before responding to the RCPT TO: command. The range is 0 through 86,400 (24 hours). The default is 60 seconds.

RESOLVERRETRYINT Statement

The RESOLVERRETRYINT statement specifies the interval (in minutes) the SMTP server should wait between attempts to resolve a host domain name.



Operands

minutes

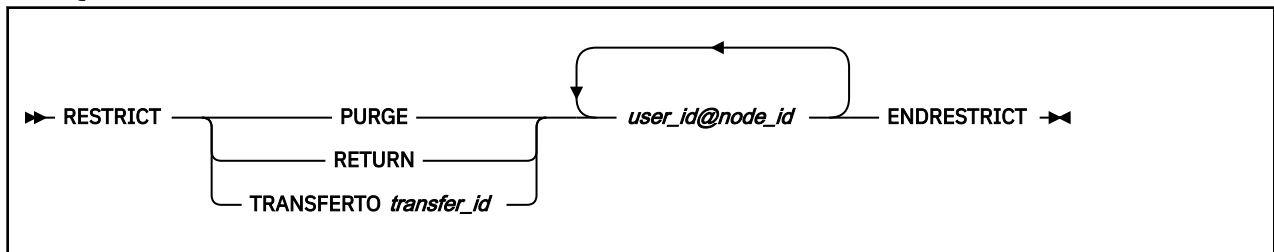
An integer in the range of 1 to 1439 (24 hours - 1 minute) that specifies the number of minutes SMTP should wait between successive attempts to resolve a host domain name. The default is 20 minutes.

RESTRICT Statement

The RESTRICT statement identifies users who may not send mail through this SMTP server. If SMTP receives a spool file from a restricted user, the spool file is purged, returned to the sender or transferred to a third party, depending on the options specified.

In addition, SMTP rejects any MAIL FROM: or RCPT TO: commands whose destinations are restricted users.

SMTP rejects only addresses that are in the restrict list; it does not check for aliases. For example, you can restrict user@host1. If host2 is an alias for host1, mail for user@host2 is not rejected, unless user@host2 is also in the restrict list.



Note: The RESTRICT statement cannot be used if the SMTP virtual machine is running as a secure gateway. Either remove or comment out the RESTRICT statements from the SMTP CONFIG file.

Operands

PURGE

Indicates that spool files from restricted users are to be purged.

RETURN

Indicates that spool files from restricted users are to be returned to each respective user.

TRANSFERTO

Indicates that spool files from restricted users are to be transferred to a specific user ID.

transfer_id

The user ID to which the spool file is transferred.

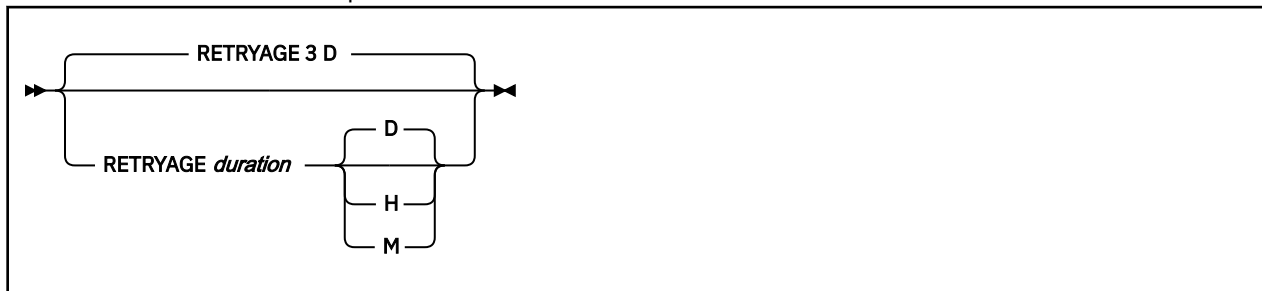
user_id@node_id

The user ID and node ID of a restricted user. This parameter can be repeated, and can include an asterisk (*) to act as a wildcard.

The ENDRESTRICT statement ends the RESTRICT statement.

RETRYAGE Statement

The RETRYAGE statement specifies the amount of time after which SMTP returns mail as undeliverable. SMTP tries to deliver mail to an inactive site at intervals determined by the RETRYINT statement. After the amount of time specified by the RETRYAGE statement has passed, SMTP returns the mail to the sender with a note that lists the recipients to which the mail could not be delivered.



Operands

duration

The amount of time (specified in days, hours, or minutes) over which SMTP is to attempt delivery of a piece of mail. If *duration* is specified in days, the range is 0 through 365; when it is given in hours, the range is 0 through 8760 (365 days). When *duration* is specified in minutes, the range is 0 through 525600 (365 days). The default is 3 days.

D

Indicates that *duration* is specified in days. This is the default.

H

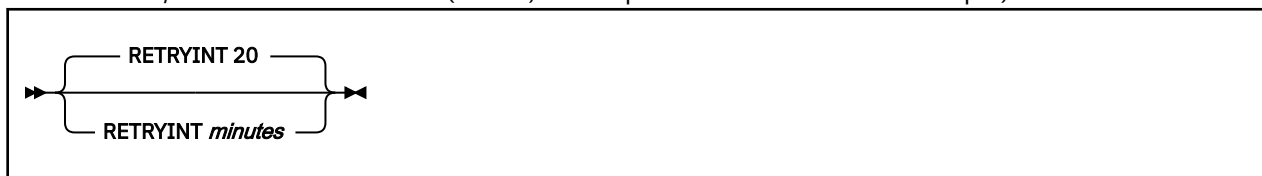
Indicates that *duration* is specified hours.

M

Indicates that *duration* is specified in minutes.

RETRYINT Statement

The RETRYINT statement specifies the number of minutes SMTP should wait between attempts to deliver mail to a TCP/IP host that is inactive (that is, not responsive to connection attempts).



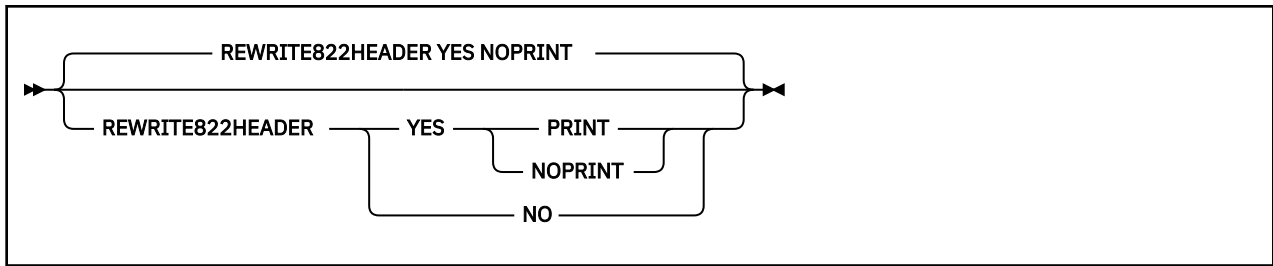
Operands

minutes

An integer in the range of 0 to 1440 (24 hours) that specifies the number of minutes SMTP should wait between mail delivery attempts to an inactive host. The default is 20 minutes.

REWRITE822HEADER Statement

The REWRITE822HEADER statement specifies the direction to SMTP to rewrite or print the RFC 822 headers of mail arriving from the RSCS side of the mail gateway.



Operands

YES

Indicates that SMTP should rewrite the RFC 822 mail headers. The YES parameter with NOPRINT is the default. SMTP uses a set of default header rewriting rules. For more information to customize the rewriting rules, see [“Customizing SMTP Mail Headers” on page 412](#).

NO

Indicates that SMTP should not rewrite the RFC 822 mail headers. This is not recommended unless all mail user agents sending mail to SMTP create RFC 822 mail headers with fully qualified domain addresses that are valid on the internet.

PRINT

Indicates that SMTP should print the RFC 822 header rewriting rules to the console.

NOPRINT

Indicates that SMTP should not print the RFC 822 header rewriting rules to the console.

RSCSDOMAIN Statement

The RSCSDOMAIN statement specifies the domain name of the RSCS network (if SMTP is running as a mail gateway). The RSCS domain name is used to rewrite the header fields of mail passing from RSCS network senders to TCP/IP network recipients.

If TCP/IP network senders qualify an RSCS network recipient's host name with a domain name that matches either the RSCS domain name (defined using this statement) or an alternate RSCS domain name (defined by the ALTRSCSDOMAIN statement), SMTP accepts and attempts local delivery of the supplied mail, due to use of the GATEWAY configuration statement and presence of SMTPRSCS HOSTINFO data. If this delivery attempt fails, the mail is rejected due to an unknown host condition.

If TCP/IP network senders do not qualify the network recipient's host name with an RSCS domain name, SMTP attempts to resolve the given host via DNS services. If a resolved destination cannot be obtained, SMTP then attempts local delivery, as described in the previous paragraph.

Note: SMTP considers the RSCS domain name BITNET to be a synonym for NETNORTH, EARN, and EARNET.

```

➤ RSCSDOMAIN — RSCS_domain_name ➤
  
```

Operands

RSCS_domain_name

A domain name, specified as a string of alphanumeric characters. For example, BITNET, VNET, and vnet.ibm.com are all valid RSCS domain names. The default RSCS domain name is a null string.

RSCSFORMAT Statement

The RSCSFORMAT statement specifies the spool file format for mail delivered to recipients on the RSCS network. This statement is valid only in GATEWAY mode.



Operands

PUNCH

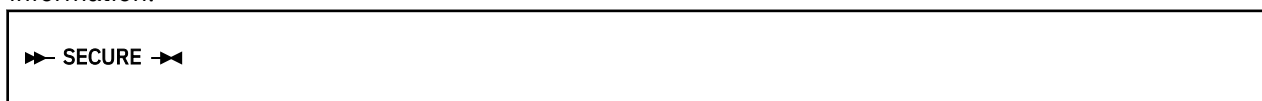
Indicates that records are folded up to 80 characters in length or fewer.

NETDATA

Indicates that records can be longer than 80 characters and arrive as MESSAGE-type records. The default format is NETDATA.

SECURE Statement

The SECURE statement specifies SMTP operates as a secure mail gateway between TCP/IP network sites and RSCS network sites. Mail is accepted by RSCS only if the user ID and node ID are included in the SMTP SECTABLE file. See [“Configuring a TCP/IP-to-RSCS Secure Mail Gateway” on page 409](#) for more information.

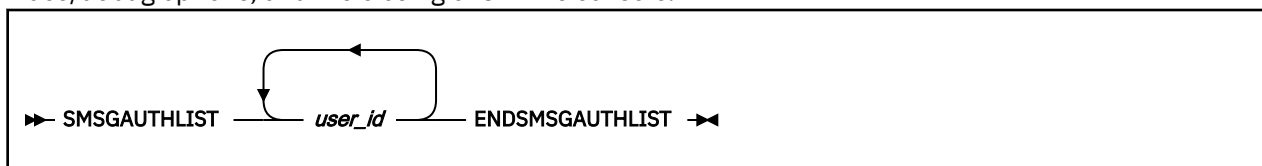


Operands

The SECURE statement has no operands.

SMSGAUTHLIST Statement

The SMSGAUTHLIST statement identifies the local and RSCS users authorized to issue privileged SMTP SMSG commands. Any VM user can issue the general usage SMTP SMSG commands, but only those users specified in the SMSGAUTHLIST statement can issue the privileged commands. Privileged SMTP SMSG commands allow the shutting down or rebooting of SMTP, the enabling or disabling of various SMTP trace/debug options, and the closing of SMTP's console.



Operands

user_id

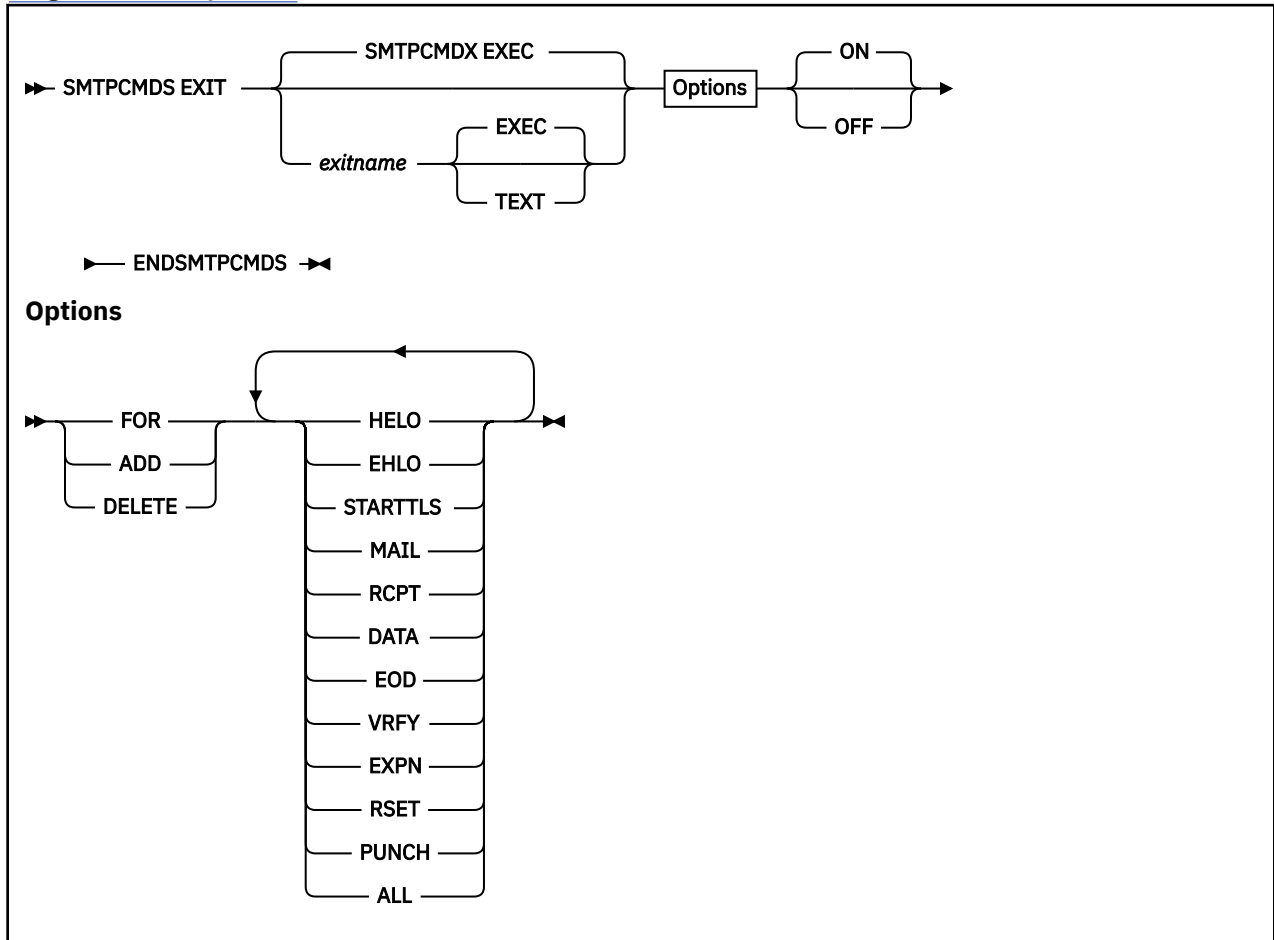
The address of a local user ID authorized to issue privileged SMTP SMSG commands.

The *user_id* parameter can be repeated. The ENDSMSGAUTHLIST statement ends the SMSGAUTHLIST statement.

For more information about the SMSG interface to SMTP, see the [“Privileged User SMSG Commands” on page 420](#).

SMTPCMDS Statement

The SMTPCMDS statement is used to define the characteristics of the SMTP command user exit and to indicate whether or not the exit is to be enabled (turned on) or disabled (turned off) when the server completes initialization. For information on using the SMTP command exit, see the [z/VM: TCP/IP Programmer's Reference](#).



Operands

exitname

The name of the exit routine associated with this command; the default exit name is **SMTPCMDX**.

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT

Indicates the exit routine name specified on this command is the name of a text deck.

FOR

Precedes the exact list of commands for which the exit is being defined.

ADD

Precedes the list of commands that are to be added to this exit definition.

DELETE

Precedes the list of commands that are to be deleted from this exit definition.

HELO

Indicates the exit routine is to be turned on or off for the HELO command.

EHLO

Indicates the exit routine is to be turned on or off for the EHLO command.

STARTTLS

Indicates the exit routine is to be turned on for the STARTTLS command.

MAIL

Indicates the exit routine is to be turned on or off for the MAIL FROM: command.

RCPT

Indicates the exit routine is to be turned on or off for the RCPT TO: command.

DATA

Indicates the exit routine is to be turned on or off for the DATA command.

EOD

Indicates the exit routine is to be turned on or off when the "end of data" condition is reached; that is, when a period (.) is received by the SMTP server after a DATA command.

VRFY

Indicates the exit routine is to be turned on or off for the VRFY command.

EXPN

Indicates the exit routine is to be turned on or off for the EXPN command.

RSET

Indicates the exit routine is to be turned on or off for the RSET command.

PUNCH

Indicates the exit routine is to be turned on or off for PUNCH processing. If the exit is enabled for this condition, it will be called when SMTP is ready to punch local mail (mail on the same node or RSCS network) to its destination.

ALL

Indicates the exit routine is to be turned on or off for all SMTP commands for which user exit capability is provided.

ON

Indicates the specified exit is being enabled (turned on) for a particular command or set of commands.

OFF

Indicates the specified exit is being disabled (turned off) for a particular command or set of commands.

Examples

- The following SMTP configuration file entry will enable the SMTP command exit routine SMTPCMDX EXEC; exit processing will be performed for only the VRFY and EXPN commands.

```
Smtpcmds
  exit smtpcmdx exec for vrfy expn on
EndSmtpcmds
```

When this entry is processed, the following text is displayed during server initialization:

```
SMTP Command Exit      : SMTPCMDX EXEC ON
- defined for          : VRFY EXPN
```

- This next entry defines the SMTP command exit routine MYEXIT TEXT for only the HELO command, but disables its use once SMTP server initialization is complete.

```
Smtpcmds
  exit myexit text for helo off
EndSmtpcmds
```

When this entry is processed, the following text is displayed during server initialization:

```
SMTP Command Exit      : MYEXIT TEXT OFF
- defined for          : HELO
```

SOURCEROUTES Statement

The SOURCEROUTES statement specifies whether the SMTP server will honor source routes. Source routes may be present on the MAIL FROM: or RCPT TO: commands processed by SMTP. With this statement, you may specify which source routes are honored.

A source route is a path that contains a routing list of hosts and a destination mailbox. The list of hosts is the *route* — information about how the mail is to arrive at its final destination; the mail is passed from one host in this list to the next until it is delivered to the intended recipient.

The specification that follows is an example of a **source route**:

```
<@HOST1,@HOST2,@HOST3:USER@HOST4>
```

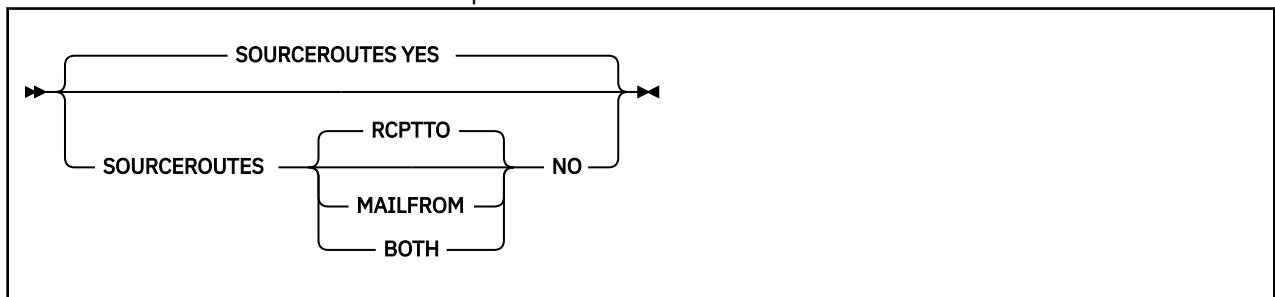
The list of hosts is HOST1, HOST2 and HOST3, and the destination is USER@HOST4.

If this sample source route is included with a RCPT TO: command, and is honored by SMTP, the mail will be sent to HOST1, then to HOST2, then to HOST3 and finally to USER@HOST4. When source routes are not honored, mail is sent directly to USER@HOST4; the list of hosts is ignored.

If this sample source route is included with a MAIL FROM: command and source routes are honored, SMTP will include its host name (for example, VMHOST1) in the path information, and will supply the following path for its MAIL FROM: command:

```
<@HOST1,@HOST2,@HOST3,@VMHOST1:USER@HOST4>
```

If such source routes are not honored, the list of hosts is removed from the mail routing path. In addition, SMTP will *not* add its host name to the path information.



Operands

YES

Indicates that source routes on the MAIL FROM: and RCPT TO: commands received from clients will be honored when mail is forwarded by SMTP. For the previous sample source route, SMTP will send the mail to HOST1 for further processing by HOST1. This is the default.

NO

Indicates that source routing is not to be honored. By default, only source routes for the RCPT TO: command will be ignored. The RCPTTO, MAILFROM or BOTH parameter can be specified with this parameter to select specific source routes to be ignored by SMTP.

Note: The NO parameter does *not* cause mail containing source routes to be rejected.

RCPTTO

Indicates that source routing is not to be honored for the RCPT TO: command. This is the default when NO is in effect. For the RCPT TO: command, any host list will be ignored and only the destination host will be used. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, SMTP will send the mail directly to USER@HOST4; the HOST1, HOST2 and HOST3 hosts will be ignored.

MAILFROM

Indicates that source routing is not to be honored for the MAIL FROM: command. When this parameter is used, the list of hosts will be removed from the mail routing path. In addition, SMTP will *not* add its host name to the path information. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, SMTP will supply the following MAIL FROM: command when mail is forwarded:

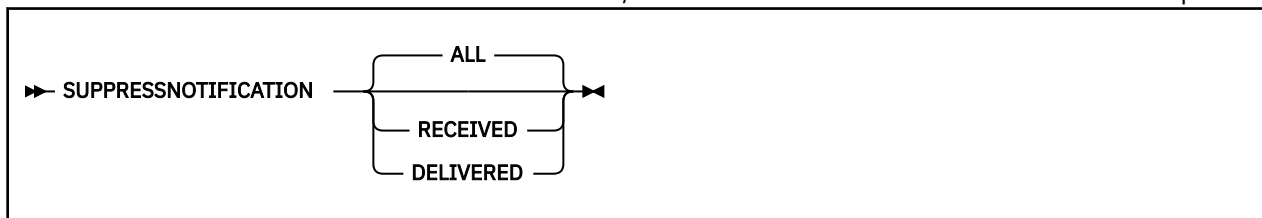
```
MAIL FROM: <USER@HOST4>
```

BOTH

Indicates that source routing is not to be honored for either of the RCPT TO: or the MAIL FROM: commands. Source routing information for these commands will be ignored as previously described.

SUPPRESSNOTIFICATION Statement

Specifies whether the SMTP server should suppress messages that are otherwise sent back to the sender when batch SMTP mail is received at the server and/or when batch SMTP mail is delivered to a recipient.

**Operands****ALL**

Suppresses all messages that are otherwise sent back to the mail sender (both the mail-received and mail-delivered messages). ALL is the default.

RECEIVED

Suppresses the mail-received messages that are otherwise sent back to the mail sender.

DELIVERED

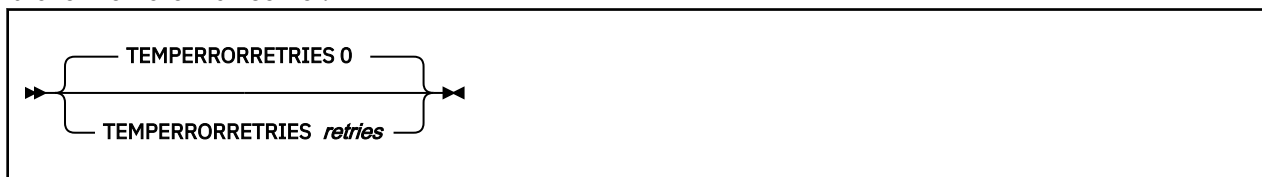
Suppresses the mail-delivered messages that are otherwise sent back to the mail sender.

Usage Notes

1. If you code multiple SUPPRESSNOTIFICATION statements in the SMTP configuration file, only the setting that you specify on the last statement is honored. For example, if a SUPPRESSNOTIFICATION DELIVERED statement follows a SUPPRESSNOTIFICATION RECEIVED statement in the SMTP configuration file, only the mail-delivered messages that are otherwise sent back to the mail sender are suppressed.

TEMPERRORRETRIES Statement

The TEMPERRORRETRIES statement specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem. Temporary problems include network congestion, network connectivity, or a broken remote mail server.



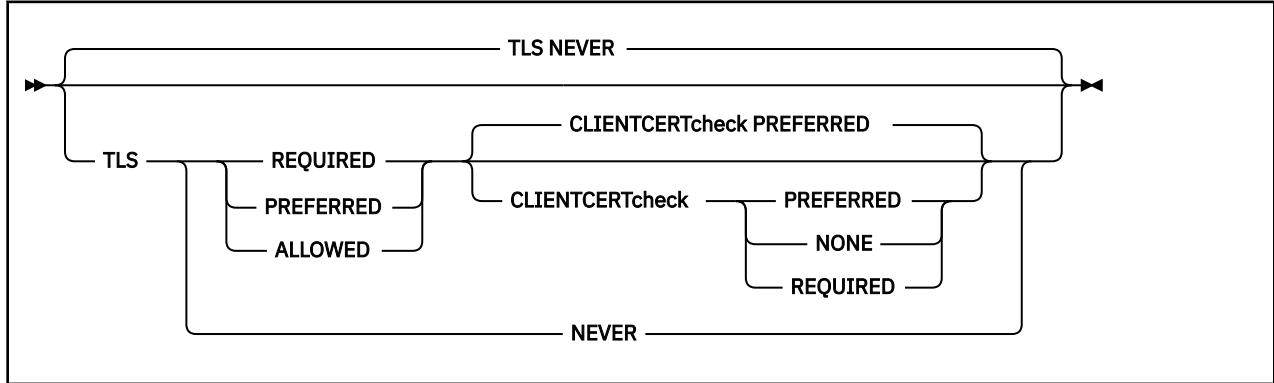
Operands

retries

The number of times mail delivery to a host with a temporary problem is retried. The default is 0, meaning that mail delivery is retried for the amount of time specified in the RETRYAGE statement. If *retries* is greater than 0, mail delivery is retried for that number of times. If delivery is still unsuccessful, the mail is returned to the sender. Change the default only when remote mail servers repeatedly terminate abnormally or hang SMTP mail transactions.

TLS Statement

The TLS statement specifies the TLS security level for the SMTP server. This statement will indicate to the server whether or not it should require a secure connection, or even allow a secure connection.



Operands

TLS REQUIRED

All connections must be secure (do not allow mail to flow across a non-secure connection).

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

TLS PREFERRED

Always try for a secure connection, but still allow mail receipt and delivery if a secure connection cannot be established.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

TLS ALLOWED

Accept secured mail, but send all mail unsecured.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

TLSSLABEL Statement

The TLSSLABEL statement specifies the TLS label to be used by the SMTP server when securing connections.

```
➤ TLSSLABEL — label ➤
```

Operands***label***

Specifies the TLS label to be used by SMTP when securing connections using TLS.

Note: The TLS label can be no more than 8 characters, and must be comprised of only uppercase, alphanumeric characters.

TRACE Statement

The TRACE statement specifies which type of tracing should be enabled during server initialization.

```
➤ TRACE — ALL — ➤
      |
      | CODEFLOW
      |
      | CONN
      |
      | DEBUG
      |
      | NOTICE
      |
      | RESOLVER
      |
      | SPL
      |
```

Operands**ALL**

Initiates tracing of all types.

CODEFLOW

Initiates tracing of SMTP code flow.

CONN

Initiates tracing of connection activity.

DEBUG

Initiates tracing of all commands and replies, and their associated connection number (this same information was previously captured using the DEBUG configuration option provided with prior releases).

NOTICE

Initiates tracing of all TCP/IP notification events that are received by the SMTP virtual machine.

RESOLVER

Initiates resolver tracing. The same information can be produced by adding the **TRACE RESOLVER** statement to the TCPIP DATA file that is read by the SMTP server virtual machine at initialization.

SPL

Initiates tracing of *SPL IUCV execution.

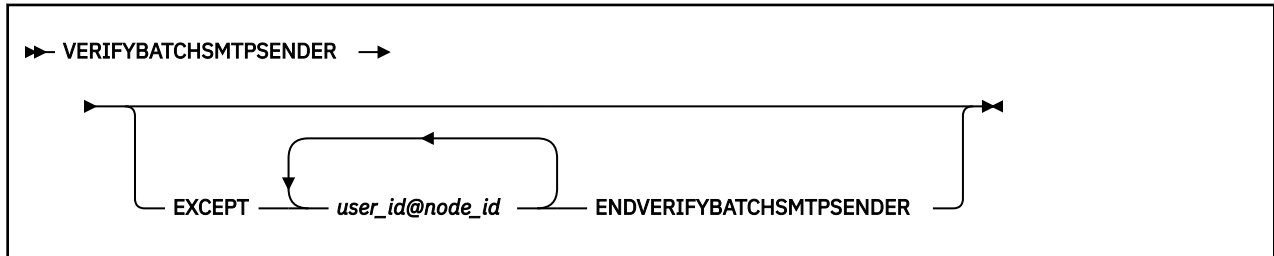
VERIFYBATCSMTPSENDER Statement

The VERIFYBATCSMTPSENDER statement specifies that MAIL FROM: information in batch SMTP spool files should be verified using available spool file TAG information. When you specify the VERIFYBATCSMTPSENDER statement, SMTP rejects batch mail that contains a MAIL FROM: address that differs from available spool file TAG information. When rejected, the MAIL FROM: command has the following error:

```
550 Spool File Origin: <userid@nodeid> does not match
    Sender's Address: <user@host>
```

You may also specify a list of users allowed to send batch SMTP mail for which the MAIL FROM: address differs from available spool file TAG information.

Rule: When VERIFYBATCSMTPSENDER is used, batch SMTP spool files cannot contain source routes in the MAIL FROM: address.



Operands

EXCEPT

Begins a list of users allowed to send batch SMTP mail for which the MAIL FROM: address differs from available spool file TAG information. If you specify a list, you must end the list with the ENDVERIFYBATCSMTPSENDER operand.

user_id@node_id

Specifies the z/VM user ID and node ID of a user that is allowed to send batch mail in which spool file TAG information differs from the sender information given in the MAIL FROM: command. The *user_id* and *node_id* must each be 8 characters or less and can each end with an asterisk (*) to act as a wildcard.

ENDVERIFYBATCSMTPSENDER

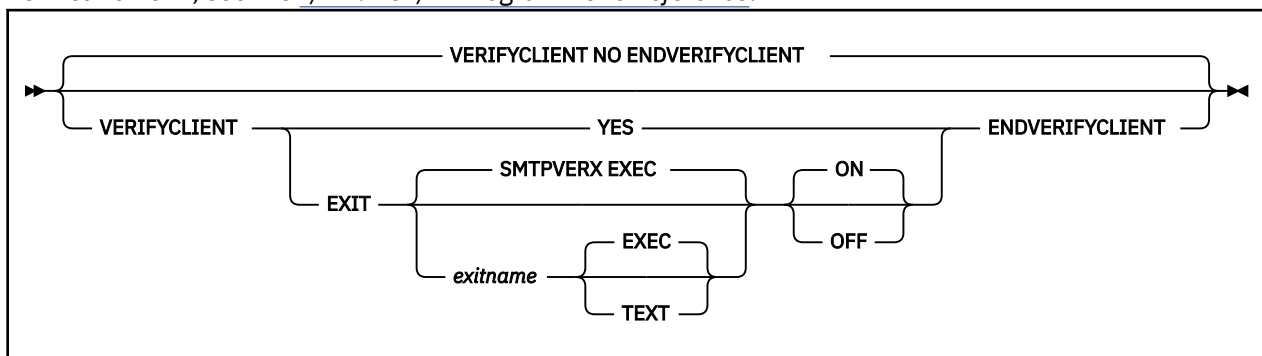
Ends the list of users started by the EXCEPT operand.

Usage Notes

1. Since the POSTMASTER commonly forwards mail on behalf of other users, z/VM users in the POSTMASTER list are automatically added to the VERIFYBATCSMTPSENDER exception list.

VERIFYCLIENT Statement

The VERIFYCLIENT statement is used to indicate to the SMTP server whether or not client verification is to be performed. Client verification can be performed using the built-in client verification function (VERIFYCLIENT YES) or using a user exit (VERIFYCLIENT EXIT). For information on using the client verification exit, see the *z/VM: TCP/IP Programmer's Reference*.



Operands

NO

Indicates that no client verification is to be performed; this is the default.

YES

Indicates verification of the client name specified on the HELO or EHLO command is to be performed using the built-in client verification function.

EXIT

Indicates a client verification exit routine is being defined.

exitname

Indicates the name of the exit routine associated with this command; the default exit name is **SMTPVERX**.

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT

Indicates the exit routine name specified on this command is the name of the text deck.

ON

Indicates the specified exit (being defined with this command) is to be enabled (on).

OFF

Indicates the specified exit (being defined with this command) is to be disabled (off).

Examples

- The following configuration file entry will enable the client verification exit routine SMTPVERX EXEC; this exit will perform all verification.

```
VerifyClient
  exit smtpverx exec on
EndVerifyClient
```

When this entry is processed, the following text is displayed during server initialization:

```
Client Verification          : Exit SMTPVERX EXEC ON
```

- This next entry defines the client verification exit routine SMTPVERX TEXT, but disables its use once SMTP server initialization is complete. Thus, client verification will not occur.


```
VerifyClient
  exit smtpverx text off
EndVerifyClient
```

When this entry is processed, the following text is displayed during server initialization:

```
Client Verification           : No (Exit SMTPVERX TEXT OFF)
```

VERIFYCLIENTDELAY Statement

The VERIFYCLIENTDELAY statement specifies the amount of time (in seconds) that SMTP will wait for the domain name system to respond during client verification. The default is 10 seconds.

If the client cannot be verified within the timeout, the mail item will be treated as if verification were not active. No comment is inserted into the mail header.



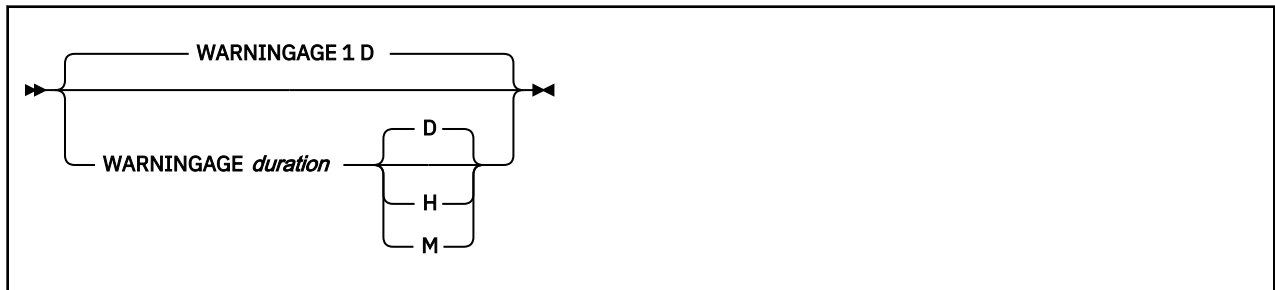
Operands

seconds

The number of seconds SMTP will wait for the domain name system to respond during client verification. The range is 0 through 86,400 seconds (24 hours). The default is 10 seconds.

WARNINGAGE Statement

The WARNINGAGE statement specifies the amount of time after which a copy of the mail is returned to the sender. The copy of the mail includes a header from SMTP that indicates the mail has been undeliverable for WARNINGAGE amount of time and that SMTP will continue to retry delivery of the mail for RETRYAGE amount of time.



Operands

duration

The amount of time (specified in days, hours, or minutes) over which SMTP is to attempt delivery of a piece of mail before sending a non-delivery warning to the sender. If *duration* is specified in days, the range is 0 through 365; when it is given in hours, the range is 0 through 8760 (365 days). When *duration* is specified in minutes, the range is 0 through 525600 (365 days). The default is 1 day.

D

Indicates that *duration* is specified in days. This is the default.

H

Indicates that *duration* is specified in hours.

M

Indicates that *duration* is specified in minutes.

When the RETRYAGE and the WARNINGAGE are equal, a warning is not issued to the sender. The warning is only issued if the WARNINGAGE is less than the RETRYAGE.

8BITMIME Statement

The 8BITMIME statement specifies the file name of the translation table to be used for 8-bit MIME support; the file type of this file must be TCPXLBIN. The translation table specified on this statement will be used *only* for 8-bit MIME support. Thus, if this statement is not specified, 8-bit MIME will not be supported.

```
➡ 8BITMIME — filename ➡
```

Operands

filename

The file name of the translation table to be used for 8-bit MIME support.

For more information, see [Chapter 19, “Using Translation Tables,”](#) on page 667.

Note: Changing the 8-bit MIME translation table could affect mail for which delivery is pending. When SMTP is restarted, notes that require 8-bit MIME support *and* are destined for non-RSCS network recipients will be processed using the new translation table.

Configuring the Server for Secure SMTP

The SMTP virtual machine may be configured by your TCP/IP administrator for Secure SMTP. This would mean that when sending and receiving mail across a TCP/IP network, SMTP will attempt to use secure connections. The method used for securing SMTP connections is known as Transport Layer Security (TLS) and is documented in RFC 3207 (SMTP Extension for Transport Layer Security). When the server is configured to support TLS, a negotiation will take place between the SMTP client and the SMTP server to secure the connection, and all data that is sent across that connection is encrypted using Secure Socket Layer (SSL).

If you choose to configure SMTP to take advantage of Secure SMTP, you will need to do the following:

1. Verify that the z/VM system that your SMTP server is running on has been configured to run with a Secure Socket Layer (SSL) server
2. Verify that you have a certificate in the certificate database for use by the SMTP server and that you know the certificate label name. For information about certificates and labels, see [Chapter 15, “Configuring the SSL Server,”](#) on page 453.
3. Add a TLS statement to the SMTP server configuration file to specify that TLS security is desired and to specify the level of client certificate checking. For more information about the TLS statement, see [“SMTP Server Configuration File Statements”](#) on page 378.
4. Add a TLSLABEL statement to the SMTP server configuration file to specify the name of the TLS label (as determined in step 2 above) to be used by the SMTP server when securing connections. The TLSLABEL statement is required for any TLS security setting other than TLS NEVER. For more information about the TLSLABEL statement, see [“SMTP Server Configuration File Statements”](#) on page 378.
5. In order to ensure that the SSL server initializes prior to the SMTP server being initialized, add an SSLSERVERID statement to the TCP/IP configuration file specifying the userid of the SSL server.

SMTP Server Exits

Several SMTP server exits are supported that allow for greater control over each piece of mail that is processed by the SMTP server. These exits are:

- the client verification exit, which might be used to reject mail from a particular host, designate certain trusted sites as “verified” but perform validation on all others, or control which users can use a particular SMTP server.
- the mail forwarding exit, which could be used to disallow forwarding of mail from a known sender of “junk” mail, intercept mail from specific clients and forward that mail to a local VM user ID for further analysis, or restrict the ability to forward mail to a particular set of hosts.
- the SMTP command exit, which allows control over specific SMTP commands. This exit might be used to reject particular SMTP commands, handle the delivery of local mail in a specific manner, or screen and reject mail based on content. For example, you may not want your server to support the VRFY and EXPN commands.

Note: The SMTP server answers to the EXPN and/or VRFY commands. The EXPN command can be used to find the delivery address of mail aliases, or even the full name of the recipients, and the VRFY command may be used to check the validity of an account. Your mailer should not allow remote users to use any of these commands, because it gives them too much information.

Prior to customizing any of these server exits, ensure that you have reviewed the exit limitations and customization recommendations presented in [“Customizing Server-specific Exits”](#) on page 49.

For more information on how to effectively use the exit routines mentioned above, see *z/VM: TCP/IP Programmer's Reference*, SC24-6332.

Configuring a TCP/IP-to-RSCS Mail Gateway

You can configure the SMTP virtual machine with the GATEWAY option to run as a mail gateway between TCP/IP network users and users located on an RSCS network attached to the local host. [Figure 3](#) on page 407 shows the SMTP virtual machine configured as a mail gateway.

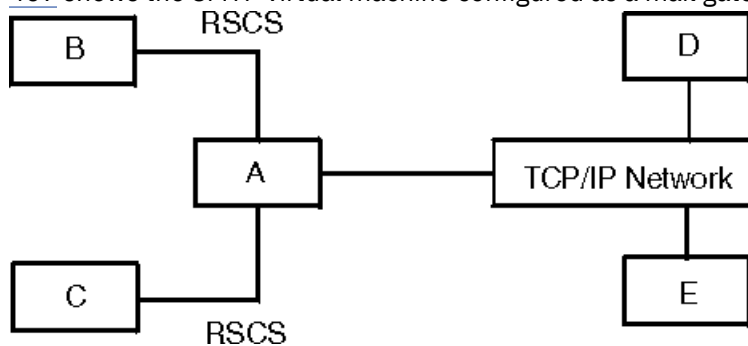


Figure 3. The SMTP Virtual Machine Configured as a Mail Gateway

Host

Description

A

The local VM host, running both TCP/IP and RSCS.

B and C

The hosts attached to host A through an RSCS network.

D and E

The 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 or SENDFILE commands, or using OfficeVision/VM. For more information, see the *z/VM: TCP/IP User's Guide*. The following steps describe how to configure a TCP/IP-to-RSCS mail gateway.

1. Update the SMTP configuration file to include the GATEWAY, RSCSDOMAIN, and RSCSFORMAT statements. Specify the GATEWAY, RSCSDOMAIN, and RSCSFORMAT options in the configuration file.
2. Enter the SMTPRSCS command. See [“SMTPRSCS Command” on page 408](#) for a description of the command.

SMTPRSCS Command

```
➤ SMTPRSCS — filename filetype filemode ➤
```

Purpose

This command creates the RSCS host table file, SMTPRSCS HOSTINFO. After the file is created, copy it to an SMTP server visible minidisk, such as TCPMAINT 198.

Operands

filename filetype filemode

The file identifier of the RSCS configuration file, for example, RSCS CONFIG A. The command requires a file name, file type and file mode.

Usage Notes

- If the local system's RSCS configuration file does not explicitly specify all of your RSCS network's nodes (for example, if it uses a default ROUTE * statement), you should either obtain the RSCS configuration file that does contain all of the RSCS network nodes and use that file as the SMTPRSCS command input, or create a local RSCS configuration file with ROUTE statements that do identify all of your RSCS network's nodes and use that as input to the SMTPRSCS command.

In order for an SMTP to RSCS gateway to work correctly, the SMTPRSCS HOSTINFO file must contain all RSCS nodes that SMTP mail could be destined for.

Perform the following on each RSCS node that sends mail through the SMTP virtual machine on a remote gateway node.

1. For each system running z/VM 7.3 or higher, place a TCPIP DATA file on each non-gateway VM system, on visible file space (for example the 190 disk). This DATA file must contain a SMTPSERVERID statement that identifies the user ID and RSCS node of your SMTP gateway. Copy the SMTPQUEUE EXEC to each system on user visible file space as well.
2. For each non-gateway system running VM at a level prior to z/VM 7.3, if you have a previously defined SMTP mail gateway, and you installed the necessary files on the non-gateway VM systems, then no action is necessary on those systems.

If you never installed the TCP/IP specific NOTE and SENDFILE functions on the non-gateway VM systems, then you need to do the following:

- a. Download the Mail Gateway package, MAILGATE VMARC from IBM: TCP/IP for z/VM and VM/ESA. The VM TCP/IP home page provides access to this package and instructions on how to extract files from the VMARC archive. Once the individual files are extracted, use the instructions to install and configure the gateway code (in the MAILGATE README file). The code you install is part of VM TCP/IP 2.4.0 (the NOTE and SENDFILE functions).
- b. Copy the SMTPQUEUE EXEC to each VM system.
- c. Copy the gateway's SMTPRSCS HOSTINFO file to each system. See the SMTPRSCS command above (step [“2” on page 408](#)) for details on creating this file.

Configuring a TCP/IP-to-RSCS Secure Mail Gateway

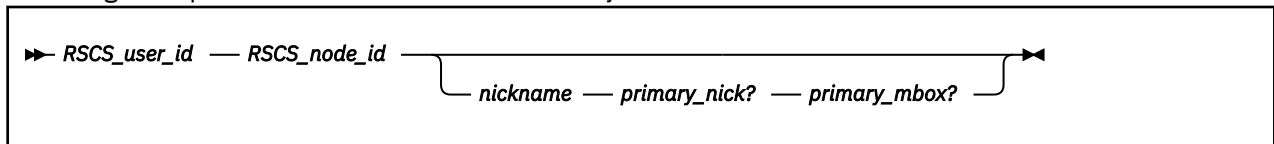
The SMTP virtual machine can be configured with the SECURE statement to run as a secure mail gateway between TCP/IP network users and users located on an RSCS network attached to the local host. For information about how to set up a mail gateway, see [“Configuring a TCP/IP-to-RSCS Mail Gateway”](#) on page 407.

To enable the SMTP Secure Gateway mode, you must add the SECURE statement to the SMTP CONFIG file. When operating in Secure Gateway mode, only those RSCS addresses in the SMTP Security Table are authorized to send or receive mail. In addition, source routing is disabled to prevent the gateway from relaying mail to unauthorized users.

SMTP rejects mail to or from an unauthorized RSCS user. If the mail is from the TCP/IP network, SMTP rejects the RCPT TO: command with the error 550 `User is not a registered gateway user`. If the mail is from the RSCS network, SMTP rejects the MAIL FROM: command with the error 550 `User is not a registered gateway user`, and includes the file SECURITY MEMO as an explanation. For more information, see the examples of rejected mail and the sample SECURITY MEMO file on [“Operands”](#) on page 409.

Creating an SMTP Security Table

Create a file called SMTP SECTABLE that contains a list of RSCS users who are authorized to use the gateway. This file can have either fixed or variable length records of up to 255 characters in length. Records whose first non-blank character is an asterisk (*) are treated as comments and are ignored. The following example shows the format of the security table.



Operands

RSCS_user_id

The RSCS user ID of the authorized user.

RSCS_node_id

The RSCS node ID of the authorized user.

nickname

The name by which the RSCS user is known on the TCP/IP side of the gateway.

primary_nick?

A primary nickname indicator, specified as **Y** or **N**. If **Y**, then mail addressed to *nickname@smtp-gateway* is automatically forwarded to *RSCS_user_id* at *RSCS_node_id*. Each nickname can only have one *primary_nick* record set to **Y**.

primary_mbox?

A primary mailbox indicator, **Y** or **N**. If **Y**, then mail from *RSCS_user_id* at *RSCS_node_id* is converted to *nickname@smtp-gateway* before it is sent to the TCP/IP recipient. Each *RSCS_user_id*, *RSCS_node_id* pair can have only one *primary_mbox?* record.

A sample security table file is supplied on the TCPMAINT 591 disk as SMTPSECT SAMPTABL. Your customized table should be stored on the TCPMAINT 198 disk as file SMTP SECTABLE.

The following is an example of an SMTP SECTABLE security table.

```

* Records for Jane Doe, within IBM
JDOE  ALMADEN
JDOE  AUSTIN
* Records for John Smith, within IBM
SMITH RALEIGH  JOHNNY Y  N
SMITH YORKTOWN JOHNNY N  Y
SMITH DALLAS   JOHNNY N  N
SMITH RALEIGH  JSMITH Y  Y

```

For example, mail sent from the following RSCS network addresses through the SMTP gateway is rewritten to the following TCP/IP network addresses. Assume the host name of the gateway is SMTP-GATEWAY.IBM.COM.

RSCS Address	TCP/IP Address
JDOE at ALMADEN	JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM
JDOE at AUSTIN	JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM
SMITH at RALEIGH	JSMITH@SMTP-GATEWAY.IBM.COM
SMITH at YORKTOWN	JOHNNY@SMTP-GATEWAY.IBM.COM
SMITH at DALLAS	JOHNNY%DALLAS@SMTP-GATEWAY.IBM.COM

Mail sent from the TCP/IP network to the following TCP/IP network addresses, is forwarded to the following RSCS network addresses. Assume the host name of the gateway is smtp-gateway.ibm.com.

TCP/IP Address	RSCS Address
JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM	JDOE at ALMADEN
JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM	JDOE at AUSTIN
JSMITH@SMTP-GATEWAY.IBM.COM	SMITH at RALEIGH
JOHNNY@SMTP-GATEWAY.IBM.COM	SMITH at RALEIGH
MITH%DALLAS@SMTP-GATEWAY.IBM.COM	SMITH at DALLAS

A sample security memo file is supplied on the TCPMAINT 591 disk as SMTPMEMO SAMPLE. Your customized memo should be stored on the TCPMAINT 198 disk as file SECURITY MEMO. The supplied sample memo file contains the following text:

The mail you sent to this SMTP gateway cannot be delivered because you are not a registered user of this gateway. Contact your local administrator for instructions on how to be authorized to use this SMTP gateway.

The following is an example of rejected mail that was sent to an unregistered RSCS user.

```
Date: Fri, 9 Oct 92 10:55:59 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 received negative reply from host:
SMTP-GATEWAY
550 User 'MATT@SMTP-GATEWAY' is not a registered gateway user

    ** Text of Mail follows **
Date: Fri, 9 Oct 92 10:55:56 EDT
From: <DANIEL@VM1.ACME.COM>
To: <MATT@SMTP-GATEWAY.IBM.COM>
Subject: Lunch

Matt,

Do you have time to meet for lunch next week? I want to discuss the
shipment of ACME iron birdseed.

Daniel
```

The following is an example of rejected mail that was sent from an unregistered RSCS user.

```
Date: Fri, 9 Oct 92 11:35:18 EST
From: <SMTP@SMTP-GATEWAY.IBM.COM>
To: <MATT@SMTP-GATEWAY.IBM.COM>
Subject: Undeliverable Mail

Unable to deliver mail to some/all recipients.
550 MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
550-User 'MATT@SMTP-GATEWAY' is not a registered gateway user.
550-
550-The mail you sent to this SMTP gateway cannot be delivered because
550-you are not a registered user of this gateway. Contact your local
550-administrator for instructions on how to be authorized to use this
```

550-SMTP gateway.

```

      ** Text of Mail follows **
HELO SMTP-GATEWAY.IBM.COM
MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
RCPT TO:<DANIEL@VM1.ACME.COM>
DATA
Date: Fri, 9 Oct 92 11:34:17 EST
From: <MATT@SMTP-GATEWAY.IBM.COM>
To: <DANIEL@VM1.ACME.COM>
Subject: Awaiting your message

```

Daniel,

When are you going to contact me about the iron birdseed and giant electromagnet that I ordered? I would like to meet with you soon.

Matt

.
QUIT

Defining Nicknames and Mailing Lists Using the SMTP NAMES File

You can use the SMTP NAMES file to set up mail aliases, forwarding, and distribution lists.

Restriction: SMTP NAMES is not like a CMS *userid* NAMES file and supports the following three nickname types only (you cannot use other tags such as TCPADDR).

- Mail aliases. You can use the :NICK. tag and the :USERID. tag to set up a simple mail alias, which is simply a different nickname for a userid on the same host.

Example: The entry:

```
:nick.brat :userid.BART
```

in the SMTP NAMES file on a TCP/IP host *ourvm.our.edu* causes all mail addressed to *brat@ourvm.our.edu* to be delivered to user ID BART on that host. *brat* becomes a mail alias for BART.

- Mail forwarding. You can use the :NICK. tag with the :USERID. and :NODE. tags to set up mail forwarding. This coding defines a nickname on the local host, and any mail addressed to that nickname is forwarded to the address specified by the :USERID. and :NODE. tags.

Example: The entry:

```
:nick.homer :userid.HOMER :node.NEWVM
```

in the SMTP NAMES file on TCP/IP host *ourvm.our.edu* (also known as OURVM on an RSCS network), causes all mail addressed to *homer@ourvm.our.edu* to be forwarded to HOMER at NEWVM on the RSCS network.

- Mail distribution lists. You can use the :NICK. tag with the :LIST. tag to set up a distribution list. This coding defines a nickname on the local host, and any mail addressed to that nickname is sent to every recipient in the list. The list can include other nicknames defined in SMTP NAMES.

Example: The entry:

```

:nick.princes
:list.hal charles hamlet charming
      andrew at buc.acme.com  albert at win.ibm.com

```

in the SMTP NAMES file on TCP/IP host *ourvm.our.edu* causes all mail addressed to *princes@ourvm.our.edu* to be sent to each recipient in the distribution list.

The nickname label can be the same as one of the user IDs that are in the list defined by that nickname. The user ID within the list will be treated as a recipient.

Example: The entry:

```
:nick.joe
:  list.tom joe
```

in the SMTP NAMES file on TCP/IP host *ourvm.our.edu* causes all mail addressed to *joe@ourvm.our.edu* to be sent to each recipient in the distribution list. Thus, a copy is sent to *tom@ourvm.our.edu* and to *joe@ourvm.our.edu*.

Customizing SMTP Mail Headers

Electronic mail has a standardized syntax for text messages that are sent across networks. The standard syntax is described in RFC 822. Messages have an envelope and content. Fields in the envelope are in a rigid format and referred to as *headers*. Envelopes contain all necessary information to accomplish transmission and delivery of the message content.

The RFC 822 standard does not dictate the internal formats used at specific sites. IBM function level 730 allows specific sites to customize the SMTP mail headers with the REWRITE822HEADER statement.

You can use the REWRITE822HEADER statement to control the way SMTP performs a rewrite of the RFC 822 mail headers. Mail headers are passed from the local system or RSCS network to the TCP/IP network. Mail headers passing from the TCP/IP network to the local system or RSCS network are not affected. Mail envelope headers are also not affected. Only the addresses under certain RFC 822 header fields can be subject to the header rewriting rules.

The header fields affected by the REWRITE822HEADER statement are:

Fields

Description

From

The originator of the message.

Resent-From

The person that forwarded the message.

Reply-To

Provides a mechanism for indicating any mailboxes to which responses are to be sent.

Resent-Reply-To

The person to whom you should forward the reply.

Return-Path

Contains definitive information about the address and route back to the originator of the message. This field is added by the mail transport service at the time of final delivery.

Sender

The authenticated identity of the AGENT that sent the message. An AGENT can be a person, system, or process.

Resent-Sender

The authenticated identity of the AGENT that has resent the message.

To

Contains the identity of the primary recipient of the message.

Cc

Contains the identity of the secondary (informational) recipients of the message.

Bcc

Contains the identity of additional recipients of the message. The content of this field is not included in copies sent to the primary and secondary recipients of the message but included in the originator's copy.

The SMTP RULES File

The SMTP RULES file contains the rewrite rules for the header addresses. You can create the SMTP RULES file during the configuration of the SMTP server to customize the address transformations to the needs of a particular site. Store the SMTP RULES file on TCPMAINT 198.

The SMTP RULES file consists of the following two sections:

FIELD DEFINITION

Contains the names of all header fields whose addresses are to be rewritten.

RULES DEFINITION

Contains the rewrite rules for the header fields.

In creating the SMTP RULES file, you must follow several syntactical conventions. The conventions are:

- The file statements are free-format. Tokens can be separated by an arbitrary number of spaces, and statements can span an arbitrary number of lines. However, you must terminate every statement with a semicolon (;).
- A character string appearing within single quotation marks ('...') is case-insensitive. For example, 'abc' represents 'abc', 'Abc', 'ABC', and so on. The use of character strings is illustrated in the following sections.
- A character string appearing within double quotation marks ("...") is case sensitive. For example, "abc" only represents "abc". It does not represent "Abc", "ABC", and so forth.

Special characters, such as @ and %, are treated the same whether enclosed by single quotation marks or double quotation marks.

- Double-hyphens ("--") are used to begin a comment. The comment extends to the end of the line.

The following sections describe the components of the SMTP RULES file.

Format of the Field Definition Section

The field definition section is the first section in any SMTP RULES file. It defines any applicable alias fields, and it is introduced by the following heading:

```
Field Definition Section
```

This section allows similar fields to be grouped under an alias or common name. This name, or alias, is used to represent the field list. You can define an arbitrary number of aliases representing a set of field lists.

An alias name can be any alphanumeric sequence of characters that is not a predefined keyword within the SMTP rules (see below). However, the alias name `DefaultFields` is treated specially by the SMTP configuration interpreter. If `DefaultFields` is defined, and if a rule is written that does not specify an associated field alias, the rules interpreter assumes that `DefaultFields` is the associated field alias.

The alias definition within this section is of the following form:

```
alias_name = alias_definition; optional comment
```

where *alias_name* is the name of the alias and *alias_definition* is an expression describing which fields are to be grouped under this alias. This expression can be as simple as a single field name. For example:

```
MyAlias = 'To';
```

The aliases can be a list or set of field names. The field names 'To' 'From' 'Cc' 'Bcc', in the following example, are part of a set of field names referenced by the alias `MyAlias`.

```
MyAlias = 'To' 'From' 'Cc' 'Bcc' ; -- first list of fields
```

You can combine field names and previously defined aliases to create a new alias. In the following example, the set of field names defined as `MyAlias` and the field names in the new alias `YourAlias`

are combined to form a third set. The new alias `TheirAlias` is the union of both aliases and contain the fields of `MyAlias` and `YourAlias`.

```
MyAlias    = 'To' 'From' 'Cc' 'Bcc';
YourAlias  = 'Errors-To' 'Warnings-To';
TheirAlias = MyAlias YourAlias;
```

In the previous example, `TheirAlias` is an alias that represents the following fields.

```
TheirAlias: 'To' 'From' 'Cc' 'Bcc' 'Errors-To' 'Warnings-To'
```

You can perform the following set algebra operations on set members of the alias to create a subset of the initial alias.

- Union operations
- Difference operations
- Intersection operations.

Union and Difference Operations: You can add or omit certain field names to a new alias of field names by using a minus sign to omit set members and an optional plus sign to include another field name. In the mathematics of sets, when you add together two or more sets, they form a union. When set members are omitted, the remaining set is created by the difference operation. In the following example `HerAlias` and `HisAlias` are defined. The alias `HisAlias` is created from the union of `TheirAlias`, `HerAlias`, and the omission of `Warning-To` and `Bcc` from the following sets.

```
HerAlias    = 'Reply-To' 'Sender';
HisAlias     = TheirAlias - 'Warnings-To' - 'Bcc' + HerAlias;
```

In the previous example, `HisAlias` is an alias that represents following fields.

```
HisAlias: 'To' 'From' 'Cc' 'Errors-To' 'Reply-To' 'Sender'
```

Intersection Operations: In addition to the union and difference operations previously shown, a field definition can include an intersection operation. When the intersection operation is applied to two field expressions, the resulting set contains the fields common to both. In the following example, `MyAlias` and `YourAlias` are defined. The alias `OurAlias` is created from the intersection of `MyAlias` and `YourAlias`. The asterisk (*) is the intersection operator.

```
MyAlias    = 'Bcc' 'Cc' 'From' 'Reply-To';
YourAlias   = 'Resent-From' 'Cc' 'Sender' 'To' 'Bcc';
OurAlias    = MyAlias * YourAlias; -- the intersection
```

In the previous example, `OurAlias` represents the following fields.

```
OurAlias: 'Bcc' 'Cc'
```

In the following complex example, `TheirAlias` is created from the intersection of `YourAlias` with the sum of `MyAlias` plus `Resent-From`.

```
TheirAlias = (MyAlias + 'Resent-From') * YourAlias;
```

In the previous example, `TheirAlias` represents the following fields.

```
TheirAlias: 'Bcc' 'Cc' 'Resent-From'
```

The parentheses within the definition of `TheirAlias` perform the same functions as in algebra. Field expressions are evaluated from left to right, but the intersection operation has greater priority than union and difference operations. If parentheses were not used in the definition of `TheirAlias`, the result would be:

```
TheirAlias: 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
```

Format of the Rule Definition Section

The rule definition section is the next section in any SMTP RULES file. It contains the header rewriting rules that define the intended address transformations, and it is introduced by the following heading.

Rule Definition Section

The rewrite rules are given using a simple, free-format pattern matching language. The basic form of each rule is:

```
alias :before-address-pattern => after-address-pattern;
```

The alias name *alias* is an optional name representing the fields for which the rule is applicable. If the alias name *alias*: is omitted from this part of the rules, then `DefaultFields` is assumed.

The sequence of tokens that define how a particular type of address is to be recognized is the *before-address-pattern* portion of the rules definition. The sequence of tokens that define how the address is to appear after the address has been rewritten is the *after-address-pattern* portion of the rules definition. The following example is the rule for converting host names.

```
A '@' RSCSHostName => A '@' TCPHostName; -- convert host names
```

In the previous example, `A '@' RSCSHostName` is the *before-address-pattern* portion of this rule, and `A '@' TCPHostName` is the *after-address-pattern* portion. This rule specifies that the address to be rewritten has an arbitrary local name (A), an at-sign, and the RSCS host name (`RSCSHostName`) of the current site. The rule also specifies that the rewritten address must contain the same arbitrary local name (A), an at-sign, and the current site's TCP host name `TCPHostName`.

Syntax Convention of the SMTP Rules

The previous example of the rewriting rules shows that you must follow several syntactical conventions. The conventions are:

- Some keywords have special meaning to the rules interpreter. For example, `RSCSHostName` keyword means the RSCS host name of the present system, and `TCPHostName` keyword means the TCP host name of the present system. For more information about valid keywords, see “Predefined Keywords within the SMTP Rules” on page 417. Some keywords, such as `TCPHostName`, have single values. Other keywords, such as `AltTCPHostName` and `AnyDomainName`, can have many possible values. To avoid ambiguity, any keyword that can have multiple values, and is used in the *after-address-pattern* of a given rule, must appear exactly once within the *before-address-pattern* of that rule. The following rule example shows a valid syntax:

```
A '@' AltTCPHostName '.' AltTCPHostName =>
A '%' TCPHostName '@' TCPHostName;
```

The following two rules have invalid syntax because the first keyword `AltTCPHostName` must be rewritten to a keyword with specific values. The `AltTCPHostName` is attempting to be rewritten to the same `AltTCPHostName` but with unknown values and becomes invalid.

```
A '@' AltTCPHostName '.' AltTCPHostName =>
A '%' AltTCPHostName '@' TCPHostName;

A '@' TCPHostName => A '@' AltTCPHostName;
```

Any rule whose *before-address-pattern* includes a keyword that has a null value is ignored during the header rewriting. Thus, if there is no `AltRSCSDomain` defined in the system configuration file, no rule that includes `AltRSCSDomain` in the *before-address-pattern* is considered during the header rewriting.

- Alphanumeric identifiers that are not within apostrophes or double quotation marks, and that are not predefined keywords, are considered wildcards in the rule statement. Wildcards represent an arbitrary (non-null) sequence of characters. The identifier `A`, in the previous rule example, is a wildcard. Thus, if host were the RSCS host name for the current site, and if `tcphost` were the TCP host name for the

current site, the previous rule example recognizes `abc@host` and `d@host` as candidates for address rewriting, and rewrites them as `abc@tcphost` and `d@tcphost` respectively. To avoid ambiguity, no two wildcards are allowed in a row, and the same wildcard cannot be used more than once, within the *before-address-pattern* of a given rule. The following rules have valid syntax:

```
A '@' B TCPHostName      => A '%' B '@' TCPHostName;
A '%' B '@' RSCSHostName => A B '@' TCPHostName;
```

The following rules have invalid syntax because the first rule has two wildcards in a row A and B. The second rule has the same wildcard A repeated:

```
A B '@' TCPHostName      => A A '%' B '@' TCPHostName;
A '%' A '@' RSCSHostName => A '@' TCPHostName;
```

- A character string appearing within apostrophes or double quotation marks tells the rules interpreter where a particular string is to appear within a header address. In the previous rule example, the '@' string in the *before-address-pattern* tells the rules interpreter that an at-sign must appear between the arbitrary character string and the RSCS host name. The '@' string in the *after-address-pattern* tells the rules interpreter that the address must be rewritten so an at-sign appears between the arbitrary string and the TCP host name. As previously mentioned, apostrophes denote case-insensitive strings, and double quotation marks denote case-sensitive strings.
- The character sequence "=>", with no spaces between the characters separates the *before-address-pattern* from the *after-address-pattern*.
- The order in which the rules are specified is important; the first rule encountered whose *before-address-pattern* matches the current address is the rule to dictate the address transformation. Once a matching rule has been found for an address, no other rule is considered.

In addition to the rules themselves, there is the capability for some simple logic to decide at system configuration time which rules within the file should become active. These conditions are specified in the form of an IF-THEN-ELSE statement as shown in the following example.

```
IF cond THEN
    statement list
ELSE
    statement list
ENDIF
```

A statement list can consist of any number of rules or nested IF statements, or both. Each IF statement, regardless of whether it is nested, must be terminated by an ENDIF keyword. As with IF statements in other languages, the ELSE clause is optional. There are only two conditions recognized by an IF statement:

- IF *predefined keyword* = '*character string*' THEN
- IF *predefined keyword* CONTAINS '*character string*' THEN ... ENDIF

The conditional operators = and CONTAINS can be prefixed by the word NOT to invert the conditions.

The *predefined keyword* must be a keyword that resolves to a single value at system configuration time. The character string in the first condition can be null. A character string cannot span more than one line.

The following is an example of the use of IF statements.

```
IF RSCSDomain = '' THEN
    A '@' AnyRSCSHostName => A '%' AnyRSCSHostName '@' TCPHostName;
ELSE
    A '@' RSCSHostName '.' RSCSDomain      => A '@' TCPHostName;
    A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
    IF RSCSDomain CONTAINS '.' THEN
        A '@' AnyRSCSHostName      =>
        A '@' AnyRSCSHostName '.' RSCSDomain;
        A '@' AnyRSCSHostName '.' RSCSDomain      =>
        A '@' AnyRSCSHostName '.' RSCSDomain;
        A '@' AnyRSCSHostName '.' AltRSCSDomain =>
        A '@' AnyRSCSHostName '.' RSCSDomain;
    ELSE
        A '@' AnyRSCSHostName      =>
        A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
```

```

A '@' AnyRSCSHostName '.' RSCSDomain =>
A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
A '@' AnyRSCSHostName '.' AltRSCSDomain =>
A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
ENDIF
ENDIF

```

Predefined Keywords within the SMTP Rules

You can use the following predefined keywords to define the header rewriting rules.

Keyword

Definition

TCPHostName

Matches the TCP host name of the system as defined by the concatenation of the HOSTNAME and DOMAINORIGIN statements in the TCPIP DATA file.

TCPHostNameDomain

Matches the domain portion of the TCP host name of the system as defined by the DOMAINORIGIN statement in the TCPIP DATA file. For example, if the TCP host name was `vm1.acme.com`, the value of TCPHostNameDomain is `acme.com`.

ShortTCPHostName

Matches the first portion of the TCP host name of the system, as defined by the HOSTNAME statement in the TCPIP DATA file. For example, if the TCP host name was `vm1.acme.com`, the value of ShortTCPHostName is `vm1`.

AltTCPHostName

Matches any alternative TCP host name of the system, as defined by ALTTCPHOSTNAME statements in the SMTP CONFIG file

RSCSHostName

Matches the RSCS host name of the system from the CMS IDENTIFY command. NJEHostName is a synonym for RSCSHostName.

AnyRSCSHostName

Matches any (unqualified) RSCS host name defined in the SMTPRSCS HOSTINFO file. AnyNJEHostName is a synonym for AnyRSCSHostName.

RSCSDomain

Matches the domain name of the RSCS network as defined by the RSCSDOMAIN statement in the SMTP CONFIG file. NJEDomain is a synonym for RSCSDomain.

AltRSCSDomain

Matches the alternative domain name of the RSCS network as defined by the ALTRSCSDOMAIN statement in the SMTP CONFIG file. AltNJEDomain is a synonym for AltRSCSDomain.

AnyDomainName

Matches any fully qualified domain name. Any host name with a period (.) is considered to be a fully qualified domain name.

SecureNickAddr

Matches an address of the form *RSCS_user_id@RSCS_node_id*, where *RSCS_user_id*, and *RSCS_node_id* are defined with a nickname in the SMTP SECTABLE file.

Note: This matches only user and node IDs that are defined with nicknames.

When SecureNickAddr is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickName with the corresponding nickname. This allows SecureNickName to be specified in the *after-address-pattern*.

SecureNickName

Matches a nickname defined in the SMTP SECTABLE file. When SecureNickName is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickAddr with the corresponding *RSCS_user_id@RSCS_node_id*. This allows SecureNickAddr to be specified in the *after-address-pattern*.

The predefined keywords defined previously can consist of any combination of uppercase and lowercase characters; the rules interpreter does not distinguish between them.

The secure keywords are only valid when SMTP is configured to be a secure gateway.

Default SMTP Rules

If the SMTP RULES file does not exist, SMTP uses a default set of rules. The default set used depends on whether SMTP is configured as a secure gateway.

SMTP Non-Secure Gateway Configuration Defaults

If SMTP is not configured as a secure gateway, SMTP uses the following default:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section

A '@' RSCSHostName => A '@' TCPHostName;

IF RSCSDomain = '' THEN
  A '@' AnyRSCSHostName => A '%' AnyRSCSHostName '@' TCPHostName;
ELSE
  A '@' RSCSHostName '.' RSCSDomain => A '@' TCPHostName;
  A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
  IF RSCSDomain CONTAINS '.' THEN
    A '@' AnyRSCSHostName =>
      A '@' AnyRSCSHostName '.' RSCSDomain;
    A '@' AnyRSCSHostName '.' RSCSDomain =>
      A '@' AnyRSCSHostName '.' RSCSDomain;
    A '@' AnyRSCSHostName '.' AltRSCSDomain =>
      A '@' AnyRSCSHostName '.' RSCSDomain;
  ELSE
    A '@' AnyRSCSHostName =>
      A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
    A '@' AnyRSCSHostName '.' RSCSDomain =>
      A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
    A '@' AnyRSCSHostName '.' AltRSCSDomain =>
      A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
  ENDIF
ENDIF

A '@' TCPHostName => A '@' TCPHostName;
A '@' ShortTCPHostName => A '@' TCPHostName;
A '@' AltTCPHostName => A '@' TCPHostName;
A '@' AnyDomainName => A '@' AnyDomainName;
A '@' B => A '@' B '.' TCPHostNameDomain;
```

SMTP Secure Gateway Configuration Defaults

If SMTP is configured as a secure gateway, SMTP uses the following default:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section

SecureNickAddr => SecureNickName '@' TCPHostName;
A '@' RSCSHostName => A '@' TCPHostName;

IF RSCSDomain NOT = '' THEN
  SecureNickAddr '.' RSCSDomain => SecureNickName '@' TCPHostName;
  SecureNickAddr '.' AltRSCSDomain => SecureNickName '@' TCPHostName;
  A '@' RSCSHostName '.' RSCSDomain => A '@' TCPHostName;
  A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
  IF RSCSDomain CONTAINS '.' THEN
```

```

A '@' AnyRSCSHostName          =>
A '@' AnyRSCSHostName '.' RSCSDomain;
A '@' AnyRSCSHostName '.' RSCSDomain =>
A '@' AnyRSCSHostName '.' RSCSDomain;
A '@' AnyRSCSHostName '.' AltRSCSDomain =>
A '@' AnyRSCSHostName '.' RSCSDomain;
ELSE
A '@' AnyRSCSHostName          =>
A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
A '@' AnyRSCSHostName '.' RSCSDomain =>
A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
A '@' AnyRSCSHostName '.' AltRSCSDomain =>
A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
ENDIF
ENDIF

A '@' TCPHostName              => A '@' TCPHostName;
A '@' ShortTCPHostName         => A '@' TCPHostName;
A '@' AltTCPHostName           => A '@' TCPHostName;
A '@' AnyDomainName            => A '@' AnyDomainName;
A '@' B                        => A '@' B '.' TCPHostNameDomain;

```

Examples of Header Rewrite Rules

The following are examples of how the default header rewriting rules affect an SMTP mail header. The example site is not a secure gateway and is configured as shown in the following example.

```

TCPHostName      = vm1.acme.com
ShortTCPHostName = vm1
AltTCPHostName   = seeds.acme.com
RSCSHostName     = vm1
RSCSDomain       = acmenet
AltRSCSDomain    = centralnet

```

In addition, assume that the following are known to be other RSCS hosts:

```

bird
iron

```

Then the following header:

```

From: abc@vm1 (Brendan Beeper)
To: "Jenny Bird" <def@bird>
Cc: ghi@iron.acmenet, j@vm1,
   k@seeds.acme.com
Subject: New Ore
Sender: "Mailing List" <owner@acmenet>
Bcc: lmno@iron.centralnet

```

is rewritten as:

```

From: abc@vm1.acme.com (Brendan Beeper)
To: "Jenny Bird" <def%bird.acmenet@vm1.acme.com>
Cc: ghi%iron.acmenet@vm1.acme.com, j@vm1.acme.com,
   k@vm1.acme.com
Subject: New Ore
Sender: "Mailing List" <owner%acmenet@vm1.acme.com>
Bcc: lmno%iron.acmenet@vm1.acme.com

```

If you change the rule before the two ENDIFs to:

```

A '@' AnyRSCSHostName '.' AltRSCSDomain =>
  '<@' TCPHostName ':' A '@' AnyRSCSHostName '.' RSCSDomain '>';

```

then the original Bcc field within our header is rewritten as:

```

Bcc: <@vm1.acme.com:lmno@iron.acmenet>

```

Note: Do not make the change shown in the previous example; it is intended only as a demonstration of the capabilities of the pattern-matching language.

Dynamic Server Operation: SMSG Interface to the SMTP Server

The VM Special Message (SMSG) command provides an interactive interface to the SMTP server to:

- Perform such general-user tasks as querying the SMTP mail delivery queues and operating statistics of the SMTP server.

For information about general-user SMSG commands, see:

- [SMSG HELP](#)
- [SMSG QUEUES](#)
- [SMSG STATS](#)

- Perform privileged system administration tasks, such as rebooting or shutting down the SMTP server and enabling or disabling various tracing and debugging options.

See [“Privileged User SMSG Commands” on page 420](#) for more information.

Note:

1. Privileged SMSG commands are accepted only from users that have been included on the SMTP server SMSGAUTHLIST configuration statement.
2. Command responses are returned to the originator of an SMSG command through the use of CP MSG commands (or CP MSGNOH commands if the SMTP server is running with CP privilege class B).

Privileged User SMSG Commands

[Table 34 on page 420](#) summarizes the privileged user SMTP SMSG commands.

Table 34. Privileged SMTP SMSG Commands

Command	Description	Location
SMSG CLOSECON	Closes the SMTP server's console log and sends it to the :Owner . identified in the DTCPARMS file.	“Privileged User SMSG CLOSECON Command” on page 421
SMSG FORWARDMAIL	Enables or disables mail forwarding, or identifies a user exit to be used to control mail forwarding.	“Privileged User SMSG FORWARDMAIL Command” on page 421
SMSG LISTMAIL	Displays the number of pieces of mail or a list of mail currently being processed by the SMTP server.	“Privileged User SMSG LISTMAIL Command” on page 423
SMSG MAILINFO	Displays general envelope information for a given piece of mail.	“Privileged User SMSG MAILINFO Command” on page 425
SMSG PURGE	Purges a single piece of mail.	“Privileged User SMSG PURGE Command” on page 426
SMSG REBOOT	Causes SMTP to do an initial program load (IPL) of CMS.	“Privileged User SMSG REBOOT Command” on page 427
SMSG REFRESH	Refresh SMTP security or nickname table information.	“Privileged User SMSG REFRESH Command” on page 427
SMSG REPROCESS	Causes a piece of mail to be reprocessed.	“Privileged User SMSG REPROCESS Command” on page 428

Table 34. Privileged SMTP MSG Commands (continued)

Command	Description	Location
MSG SHUTDOWN	Initiates SMTP server shutdown processing (in the same manner as the #CP EXTERNAL command) and logs off the server.	“Privileged User MSG SHUTDOWN Command” on page 428
MSG SMTPCMDS	Defines the characteristics of the SMTP command user exit.	“Privileged User MSG SMTPCMDS Command” on page 429
MSG SOURCEROUTES	Specifies whether or not the SMTP server is to honor source routes.	“Privileged User MSG SOURCEROUTES Command” on page 431
MSG TLS	Sets the SMTP server-wide TLS security level or overrides the current setting.	“Privileged User MSG TLS Command” on page 433
MSG TLSLABEL	Specifies the TLS label to be used by the SMTP server.	“Privileged User MSG TLSLABEL Command” on page 434
MSG TRACE	Controls the tracing activity performed by the SMTP server.	“Privileged User MSG TRACE Command” on page 435
MSG VERIFYCLIENT	Specifies whether or not client verification is to be performed.	“Privileged User MSG VERIFYCLIENT Command” on page 436

Privileged User MSG CLOSECON Command

```
➤➤ MSG — server_id — Closecon ➤➤
```

Purpose

The CLOSECON command closes the SMTP server's console log and sends it to the :Owner . identified in the DTCPARMS file.

Privileged user MSG commands are accepted only from users specified with the MSGAUTHLIST configuration statement.

Operands

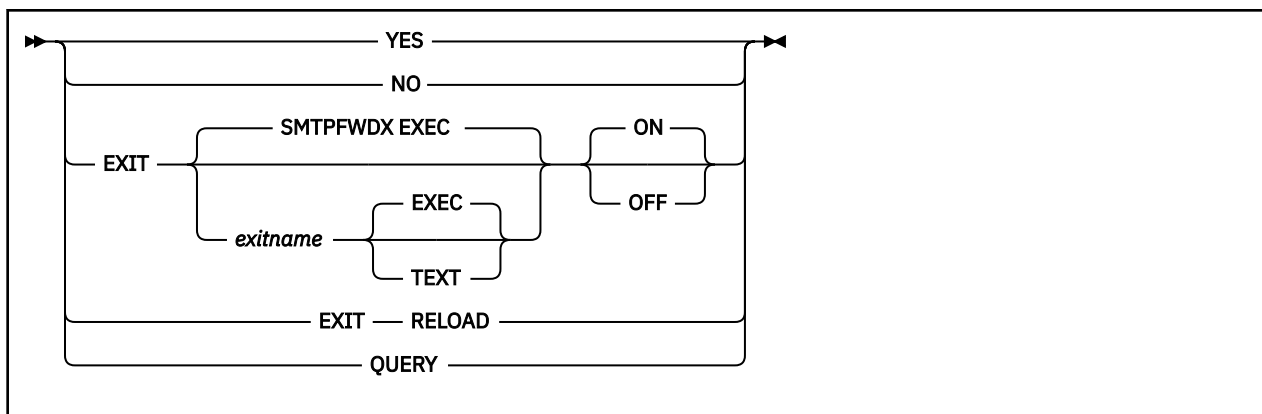
server_id

The user ID of the SMTP server virtual machine.

Privileged User MSG FORWARDMAIL Command

```
➤➤ MSG — server_id — FORWARDmail — Operands ➤➤
```

Operands



Purpose

The FORWARDMAIL command is used to enable or disable mail forwarding, to identify a user exit to be used to control such activity, and to query the current FORWARDMAIL setting. For information on using the mail forwarding exit, see the [z/VM: TCP/IP Programmer's Reference](#).

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

YES

Indicates mail forwarding is to be performed.

NO

Indicates that no mail forwarding is to be performed. When the SMTP server determines the recipient is not on the local system, the RCPT TO: command will be rejected.

EXIT

Indicates a mail forwarding exit routine is being turned on or off by this command.

exitname

The name of the exit routine associated with this command (the default exit routine name for FORWARDMAIL is SMTPFWDX).

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT

Indicates the exit routine name specified on this command is the name of a text deck.

ON

Indicates the specified exit is being enabled (turned on).

OFF

Indicates the specified exit is being disabled (turned off).

RELOAD

Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, it will be EXECLOADED into storage the next time it is executed.

QUERY

Returns current FORWARDMAIL settings; the returned response indicates whether mail forwarding is enabled or disabled. If an exit has been defined, the name of the exit is included in the response. Sample responses for several settings are shown in [Table 35 on page 423](#).

Table 35. Mail Forwarding Exit - Sample Queries

Mail Forwarding Setting	Response from Query
Mail Forwarding ON	* From SMTP: FORWARDMAIL is set to YES
Mail Forwarding OFF	* From SMTP: FORWARDMAIL is set to NO
Exit SMTPFWDX TEXT enabled	* From SMTP: FORWARDMAIL exit SMTPFWDX TEXT ON
Exit SMTPFWDX TEXT disabled	* From SMTP: FORWARDMAIL is set to YES (Exit SMTPFWDX TEXT OFF)

Examples

- The command that follows will enable the mail forwarding exit routine SMTPFWDX EXEC.

```
msg smtp forwardmail exit smtpfwdx exec on
```

The following response is displayed:

```
* From SMTP: FORWARDMAIL exit SMTPFWDX EXEC ON
```

- This next command will enable mail forwarding, and at the same time disable any mail forwarding exit that is currently in use.

```
msg smtp forwardmail yes
```

The following response is displayed:

```
* From SMTP: FORWARDMAIL is set to YES
```

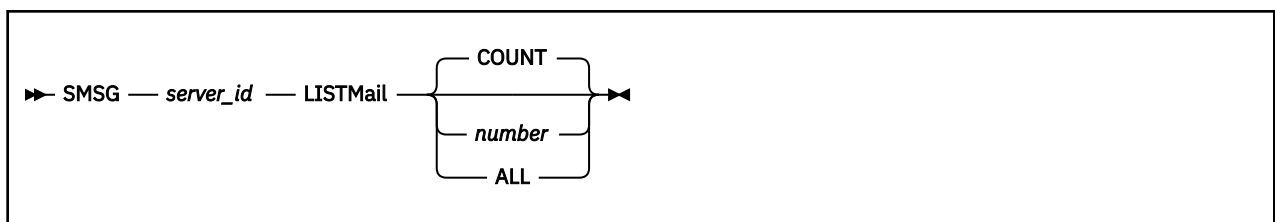
- This last command will query the current mail forwarding setting.

```
msg smtp forwardmail query
```

The following response is displayed:

```
* From SMTP: FORWARDMAIL is set to YES (Exit SMTPFWDX EXEC OFF)
```

Privileged User MSG LISTMAIL Command



Purpose

The LISTMAIL command provides the number of pieces of mail and/or a list of mail currently being processed by the SMTP server. The list will be sorted from the oldest to the newest piece of mail and will contain the following information for each piece of mail: mail ID, sender ID, Total Recipients, and Mail State.

Privileged user MSG commands are accepted only from users specified in the MSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

COUNT

Displays a count of the number of pieces of mail currently being processed by the SMTP server.
COUNT is the default.

number

The number of pieces of mail to display in the list, starting with the oldest piece of mail.

ALL

Indicates that all mail will be displayed in the list, starting with the oldest piece of mail.

Mail States

Mail can be in one or more of the following mail states:

Receiving

Mail envelope and text are currently being received by the SMTP server.

Resolving

Mail name server resolution is currently being processed.

Spooling

Mail is currently queued up for local delivery.

Waiting to Send

Mail is currently queued up for delivery to a remote host. There is currently no open connection.

Sending

Mail is currently queued up on an open connection to be sent to a remote host. The mail at the head of the queue is actively communicating with the remote host.

Waiting to Retry

Mail is currently queued up to be delivered to a remote host again because an earlier delivery attempt was unsuccessful.

Holding

Mail is currently waiting for operator intervention because local delivery was unsuccessful.

Note: Mail will remain in the *Holding* state until a REPROCESS, PURGE, REBOOT, or SHUTDOWN command is issued.

Examples

```
sm smtp listm
Ready; T=0.01/0.01 15:42:05
* From SMTP3: Total pieces of mail in process: 5

sm smtp listm count
Ready; T=0.01/0.01 15:42:50
* From SMTP3: Total pieces of mail in process: 5

sm smtp listm 3
Ready; T=0.01/0.01 15:43:22
* From SMTP3: Total pieces of mail in process: 5
* From SMTP3: Number requested: 3
* From SMTP3:
* From SMTP3:
* From SMTP3: mail Id      Sender (first 25 chars)      Total  Mail
* From SMTP3: -----      -----      -----      -----
* From SMTP3: 1            USER1@VM.WATSON.IBM.COM      8      Waiting to Retry
* From SMTP3: 2            USER3@VM.WATSON.IBM.COM     12      Waiting to Send
* From SMTP3: 3            USER@VM.WATSON.IBM.COM      3      Sending

sm smtp listm all
Ready; T=0.01/0.01 15:44:01
* From SMTP3: Total pieces of mail in process: 5
* From SMTP3: Number requested: 5
* From SMTP3:
* From SMTP3:
* From SMTP3: mail Id      Sender (first 25 chars)      Total  Mail
* From SMTP3: -----      -----      -----      -----
* From SMTP3: 1            USER1@VM.WATSON.IBM.COM      8      Waiting to Retry
* From SMTP3: 2            USER3@VM.WATSON.IBM.COM     12      Waiting to Send
* From SMTP3: 3            USER@VM.WATSON.IBM.COM      3      Sending
```

* From SMTP3: 5	jdoe@vm1.acme.com	1	Receiving+Resolving
* From SMTP3: 7	mikew@vm3.acme.com	14	Receiving

Privileged User SMSG MAILINFO Command

➤ SMSG — *server_id* — MAILInfo — *mailid* ➤

Purpose

The MAILINFO command provides general envelope information for a given piece of mail.

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

mailid

The mail ID of the piece of mail for which mail information is to be displayed. The mail ID for a piece of mail may be obtained by using the SMSG LISTMAIL command.

General Envelope Information

The MAILINFO command provides the following general envelope information for a given piece of mail:

Total Rcpts

The total number of recipients for the piece of mail.

Unresolved Rcpts

The total number of UNRESOLVED recipients for the piece of mail.

Remaining Rcpts

The total number of remaining recipients for the piece of mail.

Date Received

The date the mail was received by the SMTP server.

Time Received

The time the mail was received by the SMTP server.

Mail Size

The size of the mail in bytes.

Sender ID

The user ID of the original sender of the mail.

The following four Batch File fields are provided for a given piece of mail only if the mail was generated as the result of another virtual machine having sent a Batch SMTP (BSMTP) file to the SMTP server virtual machine:

Note:

Source User ID

The user ID of the virtual machine from which the Batch SMTP file was spooled.

Source Node ID

The node ID of the virtual machine from which the Batch SMTP file was spooled.

Source Spool ID

The spool ID of the Batch SMTP file on the virtual machine that sent the file to SMTP.

Current Spool ID

The spool ID of the Batch SMTP file on the SMTP server virtual machine.

The following general envelope information may or may not be provided, as this information is dependent upon the status of the mail:

Sender Domain

The host domain of the original sender of the mail.

Mail From String

The sender path address specified with the SMTP MAIL FROM: command for this piece of mail.

Rcpt ID

The user ID of the recipient of the mail.

Rcpt Domain

The host domain of the recipient of the mail.

Rcpt To String

The recipient path address specified with the SMTP RCPT TO: command for this piece of mail.

IP Address

The resolved recipient IP address.

Rcpt Domain, *Rcpt ID*, *Rcpt To String*, and *IP address* will be given for each recipient of the mail.

Examples

```
sm smtp mailinf 3
Ready; T=0.01/0.01 15:44:01
* From SMTP3: Mail Information
* From SMTP3: -----
* From SMTP3: Total Rcpts      : 3
* From SMTP3: Unresolved Rcpts : 0
* From SMTP3: Remaining Rcpts  : 3
* From SMTP3: Date Received    : 01/01/99
* From SMTP3: Time Received    : 15:41:52
* From SMTP3: Mail Size (bytes) : 212
* From SMTP3: Sender ID        : USER
* From SMTP3: Sender Domain     : VM
* From SMTP3: Mail From String  : <USER@VM.WATSON.IBM.COM>
* From SMTP3:
* From SMTP3: Batch File
* From SMTP3:   Source User ID   : USER
* From SMTP3:   Source Node ID   : VM
* From SMTP3:   Source Spool ID  : 1008
* From SMTP3:   Current Spool ID : 0004
* From SMTP3:
* From SMTP3: Remaining Recipients
* From SMTP3:   Rcpt ID          : user5
* From SMTP3:   Rcpt Domain      : vm1.acme.com
* From SMTP3:   Rcpt To String   : <user5@vm1.acme.com>
* From SMTP3:   IP Addresses     : 1.234.56.78
* From SMTP3:
* From SMTP3:   Rcpt ID          : user4
* From SMTP3:   Rcpt Domain      : vm1.acme.com
* From SMTP3:   Rcpt To String   : <user4@vm1.acme.com>
* From SMTP3:   IP Addresses     : 12.567.8.90
* From SMTP3:   IP Addresses     : 9.876.54.32
* From SMTP3:
* From SMTP3:   Rcpt ID          : teri
* From SMTP3:   Rcpt Domain      : vm1.acme.com
* From SMTP3:   Rcpt To String   : <teri@vm1.acme.com>
* From SMTP3:   IP Addresses     : 50D0:D2D1::9:60:60:8
* From SMTP3:   IP Addresses     : 123.456.78.9
```

Privileged User SMSG PURGE Command

➤ SMSG — *server_id* — PURge — *mailid* ➤

Operands

server_id

The user ID of the SMTP server virtual machine.

NAMES

Indicates that SMTP nickname table information should be refreshed by dynamically reading the SMTP NAMES file. The SMTP server must not be configured as a secure mail gateway.

SECTABLE

Indicates that SMTP security table information should be refreshed by dynamically reading the SMTP SECTABLE file. The SMTP server must be configured as a secure mail gateway.

Privileged User SMSG REPROCESS Command

```
➤ SMSG — server_id — REProcess — mailid ➤
```

Purpose

The REPROCESS command allows an authorized user to force a piece of mail to be reprocessed, provided it is in one of these mail states: *Waiting to Send*, *Waiting to Retry*, *Spooling*, *Sending*, and/or *Holding*.

Note: Mail that is actively communicating with a remote host cannot be reprocessed; in this case, the mail must finish communicating with the remote host before the REPROCESS command may be issued.

The REPROCESS command may be useful when a particular piece of mail cannot be delivered for some period of time, during which the IP addresses in the mail have become old or obsolete. With REPROCESS, the recipient addresses for the mail will be newly resolved and delivery of the mail will again be attempted.

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

mailid

The mail ID of the piece of mail to be processed again. The Mail ID of a piece of mail may be obtained by using the LISTMAIL command.

Examples

```
sm smtp reprocess 1
Ready; T=0.01/0.01 15:45:22
* From SMTP3: Mail Id 1 is being reprocessed.
```

Privileged User SMSG SHUTDOWN Command

```
➤ SMSG — server_id — SHutdown ➤
```

Purpose

The SHUTDOWN command initiates SMTP server shutdown processing (in the same manner as the #CP EXTERNAL command) and logs off the server.

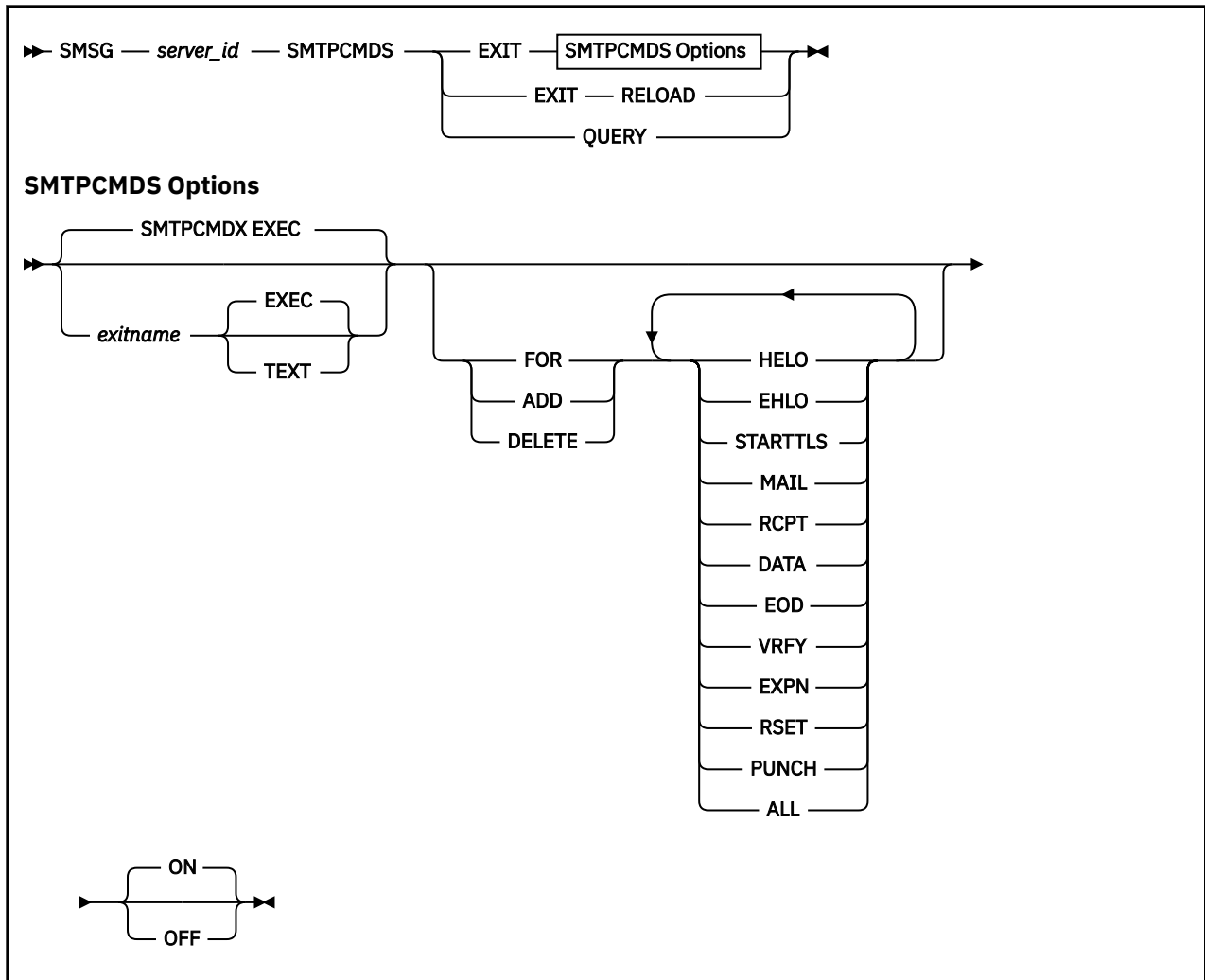
Privileged user SMSG commands are accepted only from users specified with the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

Privileged User SMSG SMTPCMDS Command



Purpose

The **SMTPCMDS** command is used to define the characteristics of the SMTP command user exit, to query those characteristics, and to indicate whether or not the exit is to be called by setting it on or off. For information on using the SMTP command exit, see the [z/VM: TCP/IP Programmer's Reference](#).

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

exitname

The name of the exit routine associated with this command (the default exit routine name for SMTPCMDS is SMTPCMDX).

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT

Indicates the exit routine name specified on this command is the name of a text deck.

FOR

Precedes the exact list of commands for which the exit is being defined.

ADD

Precedes the list of commands that are to be added to this exit definition.

DELETE

Precedes the list of commands that are to be deleted from this exit definition.

HELO

Indicates the exit routine is to be turned on or off for the HELO command.

STARTTLS

Indicates the exit routine is to be turned on or off for the STARTTLS command.

EHLO

Indicates the exit routine is to be turned on or off for the EHLO command.

MAIL

Indicates the exit routine is to be turned on or off for the MAIL FROM: command.

RCPT

Indicates the exit routine is to be turned on or off for the RCPT TO: command.

DATA

Indicates the exit routine is to be turned on or off for the DATA command.

EOD

Indicates the exit routine is to be turned on or off when the "end of data" condition is reached; that is, when a period (.) is received by the SMTP server after a DATA command.

VRFY

Indicates the exit routine is to be turned on or off for the VRFY command.

EXPN

Indicates the exit routine is to be turned on or off for the EXPN command.

RSET

Indicates the exit routine is to be turned on or off for the RSET command.

PUNCH

Indicates the exit routine is to be turned on or off for PUNCH processing. If the exit is enabled for this condition, it will be called when the SMTP server is ready to punch local mail (mail on the same node or RSCS network) to its destination.

ALL

Indicates the exit routine is to be turned on or off for all of the SMTP commands for which user exit capability is provided.

ON

Indicates the specified exit is being enabled (turned on) for a particular command or set of commands.

OFF

Indicates the specified exit is being disabled (turned off) for a particular command or set of commands.

RELOAD

Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, it will be EXECLOADED into storage the next time it is executed.

QUERY

Returns current SMTPCMDX exit settings. The returned response indicates whether the command exit is enabled or disabled. If an exit has been defined, the name of the exit is included in the response. The commands associated with the current exit state are indicated as well. Sample responses for several settings are shown in Table 36 on page 431.

Table 36. SMTP Command Exit - Sample Queries	
SMTP Command Exit Setting	Response from Query
Exit not defined	* From SMTP: SMTPCMDX exit not defined
Exit SMTPCMDX EXEC enabled for HELO, VRFY, and EXPN	* From SMTP: SMTPCMDX exit SMTPCMDX EXEC ON * From SMTP: SMTPCMDX defined for HELO VRFY EXPN
Exit SMTPCMDX EXEC disabled	* From SMTP: SMTPCMDX exit SMTPCMDX EXEC OFF * From SMTP: SMTPCMDX defined for HELO VRFY EXPN

Examples

- The command that follows will enable the SMTP command exit routine SMTPCMDX EXEC for the SMTP HELO command:

```
smsg smtp smtpcmds exit for helo
```

The following response is displayed:

```
* From SMTP: SMTPCMDX exit SMTPCMDX EXEC ON
* From SMTP: SMTPCMDX defined for HELO
```

- The next command will add the SMTP commands VRFY and EXPN to the previous exit definition:

```
smsg smtp smtpcmds exit add vrfy expn
```

The following response is displayed:

```
* From SMTP: SMTPCMDX exit SMTPCMDX EXEC ON
* From SMTP: SMTPCMDX defined for HELO VRFY EXPN
```

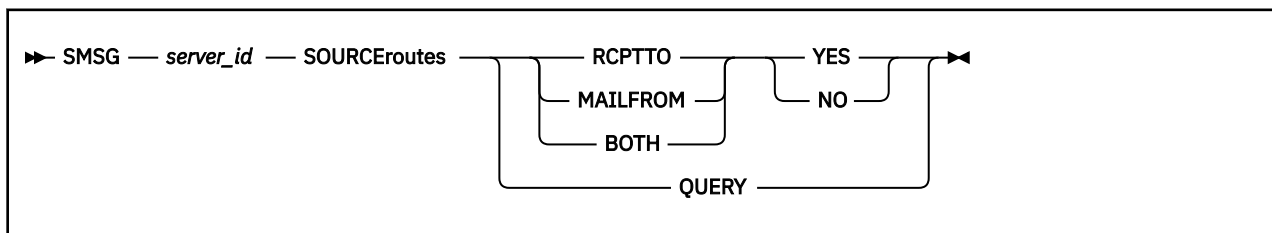
- Lastly, the command that follows will disable the SMTP commands exit:

```
smsg smtp smtpcmds exit off
```

The following response is displayed:

```
* From SMTP: SMTPCMDX exit SMTPCMDX EXEC OFF
* From SMTP: SMTPCMDX defined for HELO VRFY EXPN
```

Privileged User SMSG SOURCEROUTES Command



Purpose

The SOURCEROUTES command is used to specify whether the SMTP server will honor source routes, and to query the current SOURCEROUTES setting.

A source route is a path that contains a routing list of hosts and a destination mailbox. The list of hosts is the *route* — information about how the mail is to arrive at its final destination; the mail is passed from one host in this list to the next until it is delivered to the intended recipient.

The specification that follows is an example of a **source route**:

```
<@HOST1,@HOST2,@HOST3:USER@HOST4>
```

The list of hosts is HOST1, HOST2 and HOST3, and the destination is USER@HOST4.

If this sample source route is included with a RCPT TO: command, and is honored by SMTP, the mail will be sent to HOST1, then to HOST2, then to HOST3 and finally to USER@HOST4. When source routes are not honored, mail is sent directly to USER@HOST4; the list of hosts is ignored.

If this sample source route is included with a MAIL FROM: command and source routes are honored, the SMTP server will include its host name (for example, VMHOST1) in the path information, and will supply the following path for its MAIL FROM: command:

```
<@HOST1,@HOST2,@HOST3,@VMHOST1:USER@HOST4>
```

If such source routes are not honored, the list of hosts is removed from the mail routing path. In addition, the SMTP server will *not* add its host name to the path information.

Operands

server_id

The user ID of the SMTP server virtual machine.

YES

Indicates that client-supplied source routes on the MAIL FROM: command, the RCPT TO: command, or both will be honored when mail is forwarded by the SMTP server. The RCPT TO, MAIL FROM or BOTH parameter determines the specific source routes honored by the SMTP server.

NO

Indicates that client-supplied source routes on the MAIL FROM: command, the RCPT TO: command, or both are not to be honored when mail is forwarded by the SMTP server. The RCPT TO, MAIL FROM or BOTH parameter determines the specific source routes ignored by the SMTP server.

Note: The NO parameter does *not* cause mail containing source routes to be rejected.

RCPTTO

Indicates the processing of source routes supplied with the RCPT TO: command is to be affected. When this parameter is used and NO is specified, any host list will be ignored and only the destination host will be used. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, the SMTP server will send the mail directly to USER@HOST4; the HOST1, HOST2 and HOST3 hosts will be ignored.

When used with the YES parameter, source routes will be honored.

MAILFROM

Indicates the processing of source routes supplied with the MAIL FROM: command is to be affected. When this parameter is used and NO is specified, the list of hosts will be removed from the mail routing path. In addition, the SMTP server will *not* add its host name to the path information. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, the SMTP server will supply the following MAIL FROM: command when mail is forwarded:

```
MAILFROM: <USER@HOST4>
```

When used with the YES parameter, source routes will be honored, with the SMTP server host name included as part of the path information.

BOTH

Indicates the handling of source routes supplied for both the RCPT TO: or the MAIL FROM: commands is to be affected. Source routing information for these commands will be processed as previously described.

QUERY

Displays the current SOURCEROUTES setting. The returned response indicates whether or not source routes will be honored.

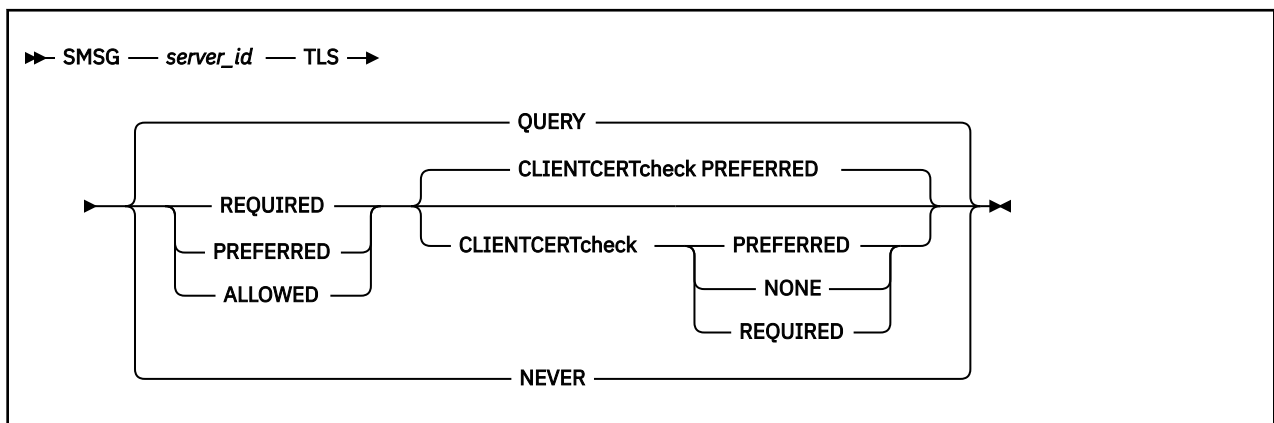
Examples

- The command that follows will disable source routes for the RCPT TO: command. Source routes will not be honored and will be ignored.

```
msg smtp sourceroutes rcptto no
```

The following response is displayed:

```
* From SMTP: RCPTTO SOURCEROUTES is set to NO
```

Privileged User MSG TLS Command**Purpose**

Dynamically sets the SMTP server-wide TLS security level or overrides the current setting. The TLS security level indicates to the server whether or not it should require a secure connection, or whether or not it should even allow a secure connection.

Operands**TLS QUERY**

Query the TLS security settings.

TLS NEVER

The STARTTLS command is not advertised and secure connections are not supported.

TLS REQUIRED

All connections must be secure (do not allow mail to flow across a non-secure connection).

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

TLS PREFERRED

Always try for a secure connection, but still allow mail receipt and delivery if a secure connection cannot be established.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

TLS ALLOWED

Only attempt to establish a secure connection if a STARTTLS is received.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

Usage Notes

The CERTFULLCHECK and CERTNOCHECK operands are deprecated.

Privileged User SMSG TSLABEL Command

➤ SMSG — *server_id* — TSLABEL — *label* ➤

Purpose

Dynamically specifies the TLS label to be used by the SMTP server.

Operands***server_id***

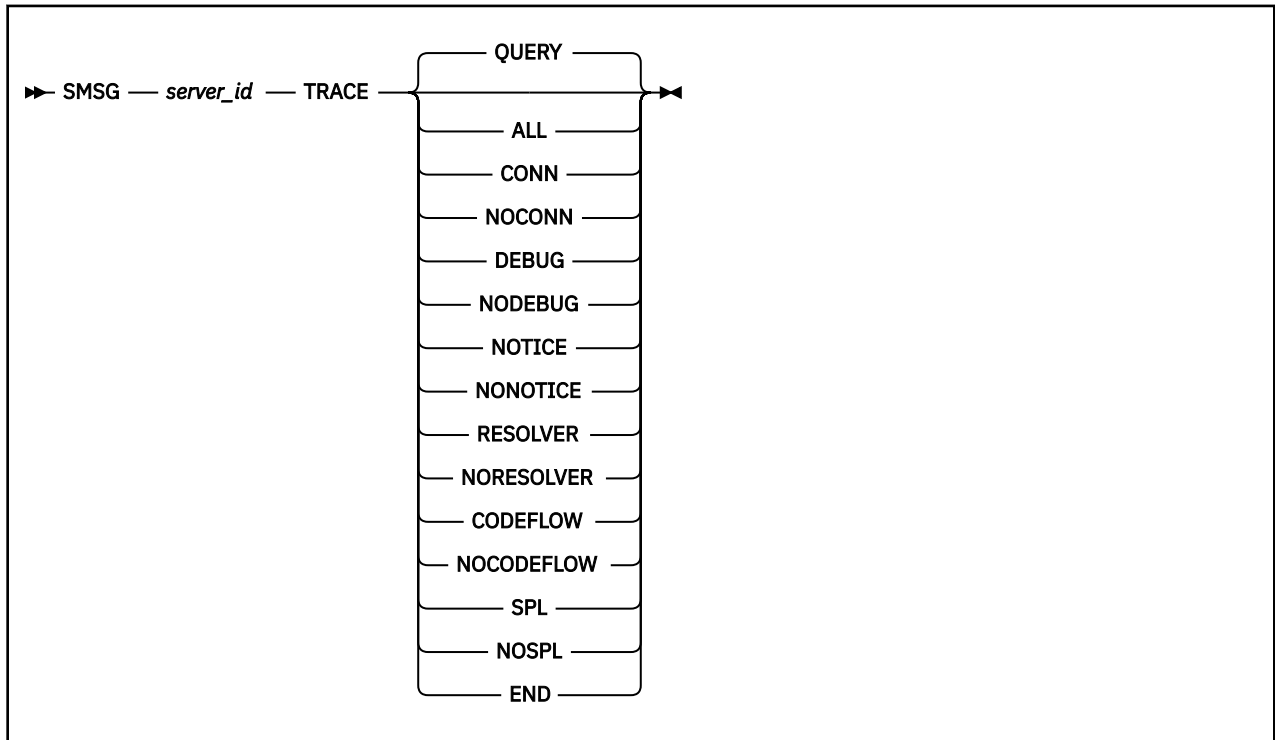
Specifies the SMTP server userid to which this SMSG command is targeted.

label

Specifies the TLS label to be used by SMTP when securing connections using TLS.

Note: The TLS label can be no more than 8 characters, and must be comprised of only uppercase, alphanumeric characters.

Privileged User SMSG TRACE Command



Purpose

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

QUERY

Reports the tracing activities that are currently in effect.

ALL

Initiates tracing of all types.

CONN

Initiates tracing of connection activity.

NOCONN

Terminates tracing of connection activity.

DEBUG

Initiates tracing of all commands and replies and their associated connection numbers (this is the same information that was captured using the old DEBUG configuration option).

NODEBUG

Terminates tracing of commands and replies.

NOTICE

Initiates tracing of all TCP/IP notification events that are received by the SMTP server.

NONOTICE

Terminates tracing of TCP/IP notification events.

RESOLVER

Initiates resolver tracing. This is the same as adding the TRACE RESOLVER statement to the TCPIP DATA file that is read by the SMTP server at initialization.

NORESOLVER

Terminates resolver tracing.

CODEFLOW

Initiates tracing of SMTP program code flow.

NOCODEFLOW

Terminates code flow tracing.

SPL

Initiates tracing of *SPL IUCV execution.

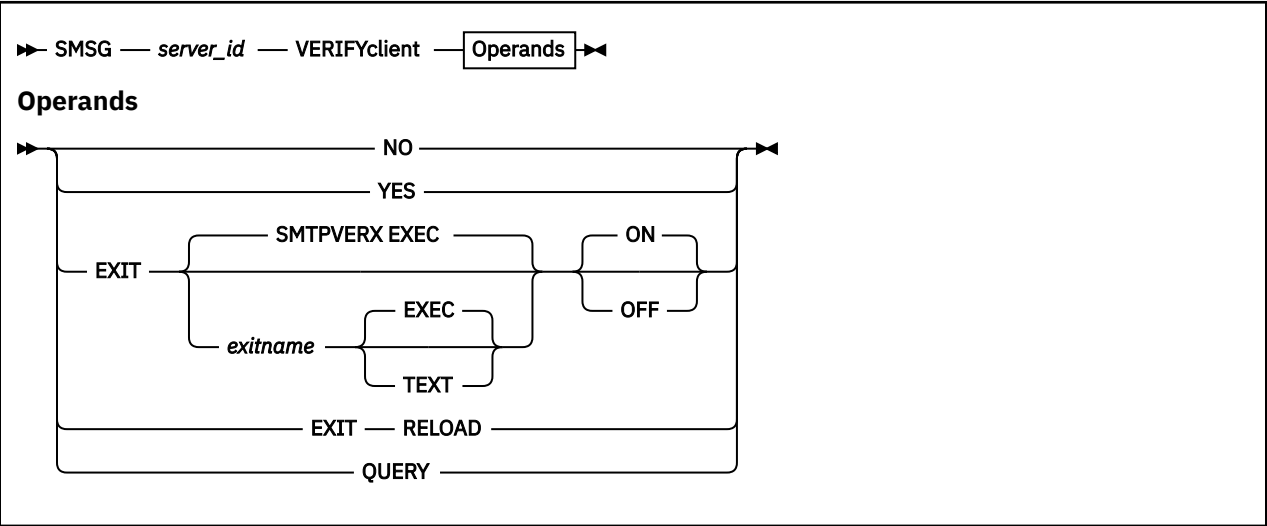
NOSPL

Terminates *SPL IUCV tracing.

END

Terminates all tracing activity.

Privileged User MSG VERIFYCLIENT Command



Purpose

The **VERIFYCLIENT** command is used to indicate whether or not client verification is to be performed, and to query **VERIFYCLIENT** settings. Client verification can be performed using the built-in client verification function (**VERIFYCLIENT YES**), or using a user exit (**VERIFYCLIENT EXIT**). For information on using the client verification exit, see the *z/VM: TCP/IP Programmer's Reference*.

Privileged user **MSG** commands are accepted only from users specified in the **MSGAUTHLIST** configuration statement.

Operands

server_id

The user ID of the SMTP server virtual machine.

NO

Indicates no client verification is to be performed.

YES

Indicates verification of the client name specified on the **HELO** or **EHLO** command is to be performed using the built-in client verification function.

EXIT

Indicates a client verification exit routine is being turned on or off by this command.

exitname

The name of the exit routine associated with this command (the default exit routine name is SMTPVERX).

EXEC

Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT

Indicates the exit routine name specified on this command is the name of a text deck.

ON

Indicates the specified exit is being enabled (turned on).

OFF

Indicates the specified exit is being disabled (turned off).

RELOAD

Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, then it will be EXECLOADED into storage the next time it is executed.

QUERY

Returns current VERIFYCLIENT settings; the returned response indicates whether client verification is enabled (on) or disabled (off). If an exit has been defined, the name of the exit is included in the response. Sample responses for several settings are shown in [Table 37 on page 437](#).

<i>Table 37. Client Verification Exit - Sample Queries</i>	
Client Verification Setting	Response from Query
Built-in function ON	* From SMTP: VERIFYCLIENT is set to YES
Built-in function OFF	* From SMTP: VERIFYCLIENT is set to NO
Exit SMTPVERX TEXT enabled	* From SMTP: VERIFYCLIENT exit SMTPVERX TEXT ON
Exit SMTPVERX TEXT disabled	* From SMTP: VERIFYCLIENT is set to NO (Exit SMTPVERX TEXT OFF)

Examples

- The command that follows will enable the client verification exit routine SMTPVERX EXEC; client verification will be performed by this exit routine.

```
smsg smtp verifyclient exit smtpverx exec on
```

The following response is displayed:

```
* From SMTP: VERIFYCLIENT exit SMTPVERX EXEC ON
```

- This next command will disable all client verification; no client verification is performed.

```
smsg smtp verifyclient no
```

The following response is displayed:

```
* From SMTP: VERIFYCLIENT is set to NO
```

- This last command will query the current client verification setting.

```
smsg smtp verifyclient query
```

The following response is displayed:

```
* From SMTP: VERIFYCLIENT is set to NO (Exit SMTPVERX EXEC OFF)
```


Chapter 14. Configuring the SNMP Servers

This chapter describes how to configure the Simple Network Management Protocol (SNMP) virtual machines. To monitor a TCP/IP VM network from your VM system, you must use NetView for z/VM.

SNMP Overview

SNMP is an architecture that allows you to manage an internet environment. You can use SNMP to manage elements such as gateways, routers, bridges, and hosts on the network. Network management stations act as clients to run the management applications that monitor the network. Network elements act as servers and contain management agents, which perform the management functions required. SNMP provides the communication between these elements and stations to send and receive information about an internet's resources.

The MIB consists of information about these resources and elements. The MIB is referred to as a database, but it actually exists as counters and temporary storage areas on most of the servers. Data in the MIB is designed according to international standards for internet management and defined according to the International Standard Organization (ISO) Abstract Syntax Notation 1 (ASN.1). For more information about the MIB, see RFC 1155.

z/VM also provides a generic SNMP subagent application that runs in a separate virtual machine and uses the Distributed Programming Interface to extend the SNMP daemon (agent) and support additional MIB variables through the use of exit routines. An exit routine is provided that can be used in conjunction with the SNMP subagent to provide RFC 1493 BRIDGE-MIB variables for the z/VM virtual switch.

More functions are available by using the SNMP daemon Distributed Program Interface (DPI). The DPI permits end users to dynamically add, delete, or replace management variables in the MIB without requiring the recompile of the local SNMP agent. DPI also allows you to generate customized TRAPs. For more information about the SNMP DPI, see [z/VM: TCP/IP Programmer's Reference](#).

In addition, a command is available to generate enterprise-specific traps in a CMS environment. See [z/VM: TCP/IP User's Guide](#) for a description of the SNMPTRAP command.

TCP/IP network management operates at the application level from one or more hosts within an internet. Each participating host or gateway runs a server program to support the SNMP functions. A network manager invokes client software at the local host computer, which has a specified client server. The client contacts a command processor or query engine and sends queries to obtain information or send commands to vary conditions for a particular managed entity such as a gateway or host TCP/IP link. Only authenticated managers can participate in this process.

Configuring the SNMP Daemon

SNMP Daemon Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the DTCPARMS file for SNMPD and SNMPSUBA 3. Create the MIB data file. 4. Configure the SNMP Daemon.

Step 1: Update PROFILE TCPIP

Include SNMPD in the AUTOLOG statement to automatically start the SNMPD virtual machine when TCPIP is invoked. Verify that the following statements have been added to PROFILE TCPIP.

```
AUTOLOG
SNMPD 0                ; SNMP daemon
```

SNMP requires that port UDP 161 be reserved for all messages sent to the VM agent. Verify that the following statements have been added to PROFILE TCPIP.

```
PORT
161 UDP SNMPD          ; SNMP daemon port for SNMP messages
```

The SNMP Query Engine and agent use raw sockets for the DPI interface and the SNMP PING function. To allow the SNMP daemon (SNMPD) to create RAW sockets, add SNMPD to the OBEY list in the PROFILE TCPIP file. Verify that the following statements have been added to PROFILE TCPIP.

```
OBEY
SNMPD
```

For more information on the OBEY list, see [“OBEY Statement” on page 597](#).

The MIB II variable sysContact identifies the contact person for this managed node. Define the contact person for this node using the SYSCONTACT statement, as shown in the following example:

```
SYSCONTACT
Andrew Ford, extension 1234
ENDSYSCONTACT
```

The MIB II variable sysLocation identifies the physical location of this managed node. Define the physical location of this node using the SYSLOCATION statement, as shown in the following example.

```
SYSLOCATION
690 Market street, bldg 100
3rd floor, room 398
ENDSYSLOCATION
```

If Bridge MIB variables are to be obtained for a virtual switch defined on the system, use an existing link or add DEVICE, LINK, START, and HOME statements to add a new interface, including the VSWITCH keyword and virtual switch name on the HOME statement. You must use a different link for each virtual switch. For more information on setting up SNMP to support BRIDGE-MIB variables, refer to [“Setting up an SNMP Subagent” on page 444](#).

Step 2: Update the DTCPARMS File for SNMPD and SNMPSUBA

When the SNMPD server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the SNMP servers are:

```
:Nick.SNMPD
:parms.
:Nick.SNMPSUBA
:parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,” on page 33](#).

Note: You should modify the DTCPARMS file for the SNMPD and SNMPSUBA servers if you:

- Activate and specify the level of tracing needed.
- Require a trace of IUCV communication to be done.
- Choose to have the SNMP daemon listen on a port other than port 161.
- Wish to have the SNMP daemon (SNMPD) automatically bring up an SNMP subagent virtual machine (for example, SNMPSUBA). See [“Setting up an SNMP Subagent” on page 444](#) for more DTCPARMS configuration information.

Step 3: Create the MIB Data File

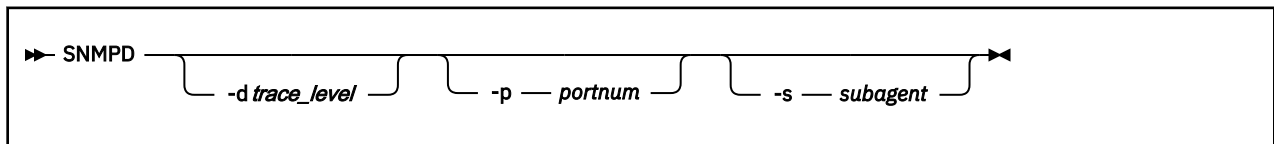
The Management Information Base (MIB) data file, `MIB_DESC DATA`, defines the short names for MIB variables. Short names are the character representation for the ASN.1 variable names. For example, `sysUpTime` is the short name for `1.3.6.1.2.1.1.3.0` (the MIB variable that stores the time since the MIB object was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP for VM ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP daemon. You can always use ASN.1 language to specify the variable names in an enterprise-specific tree (assuming that the agent supports them). You can use these short names to specify the MIB variables.

Refer to “Step 3: Create the MIB Data File” on page 448.

Step 4: Configure the SNMP Daemon

This section describes how to configure the SNMP Daemon.

SNMPD Command



Purpose

The `SNMPD` command is used to initialize a user ID to provide SNMP Daemon services.

Operands

-d *trace_level*

Specifies the level of tracing to be started. Valid values for *trace_level* are:

- 0** No tracing (default)
- 1** Trace snmpd internals
- 2** Trace external changes from egp
- 3** Trace incoming requests
- 4** Trace outgoing responses
- 5** Trace all levels

-p *portnum*

Specifies a port number on which the `SNMPD` is to listen. The default port number, if not specified, is port 161.

-s *subagent*

Specifies the userid of an SNMP subagent virtual machine that the SNMP daemon should bring up automatically (XAUTOLOG) as part of its initialization process.

TRAP Destination file

TRAPs are unsolicited messages that are sent by an SNMP daemon to an SNMP network management station. An SNMP TRAP contains information about a significant network event. The management application running at the management station interprets the TRAP information sent by the SNMP daemon.

The following TRAPs are generated by an SNMP daemon in TCP/IP. The z/VM system name is used as the community string.

- COLD_START
- AUTHENTICATION_FAILURE
- LINK_UP
- LINK_DOWN

Note: The SNMP daemon Distributed Program Interface (DPI) allows external processes (which can be running on another host) to generate TRAPs. This can allow for support of other types of TRAPs. For more information about SNMP DPI, see [z/VM: TCP/IP Programmer's Reference](#).

To use TRAPs, create a file called SNMPTRAP DEST. This file defines a list of managers to which TRAPs are sent. The following describes how to set up SNMPTRAP DEST.

1. Create a file called SNMPTRAP DEST. This file must be on a disk accessible to the SNMP agent, such as TCPMAINT 198.
2. The SNMPTRAP DEST file has a list of managers who are to receive the TRAPs, and identifies the protocol used to send TRAPs. The format of an entry in the file is:

```
manager UDP
```

The *manager* is the host that the TRAP is to be sent to. This can be a host name, or it can be the IP address of the host. The protocol must be UDP. There should be one entry in the file for each host to which you want to send traps.

3. A SNMPTRAP DEST file might contain the following:

```
124.34.216.1 UDP
Host1        UDP
```

4. Comments and sequence numbers are not allowed in the SNMPTRAP DEST file.
5. The SNMPTRAP command is available to generate enterprise-specific traps in a CMS environment. See [z/VM: TCP/IP User's Guide](#).

PW SRC File

SNMP agents are accessed by remote network management stations. These network management stations can be located anywhere in the internet. With the exception of TCP/IP, the network management stations do not have to be directly linked to the host running the SNMP agent.

To allow network management stations to send inquiries to the SNMP agent, you must create a file called PW SRC, which defines a list of community names and IP addresses that can use the services of the SNMP agent.

The community names reside in a master file. This file should be created and kept at a secure location, accessible only to a system administrator and the SNMP daemon.

The following describes how to create a community name file.

1. Create the PW SRC file on the TCPMAINT 198 minidisk accessible to the SNMP daemon.

Note: Since the PW SRC file can contain passwords, you should control access to this file if security is a concern.

2. The PW SRC file has a list of community names that can use the services of the SNMP daemon. The format of an entry in this file is:

```
community_name network_addr network_mask priv_mask
```

The *community_name* can be up to 15 characters in length. This value can contain both upper and lower case letters, however it is case sensitive. In any requests received by the SNMP agent, the *community_name* must match the *community_name* specified in PW SRC exactly, including correct case.

Note: The *priv_mask* can be used to start or stop tracing of events in the SNMP agent. If *priv_mask* is omitted, the following default mask is assumed:

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
```

3. The following shows a sample PW SRC file containing multiple entries.

```
* This is a sample PW SRC file. First we see some sample entries.
* NetView on the mainframe passes Uppercase community names from the
* command line, that is why we have the NVTEST entry as a sample.
*
* community_name netw_addr netw_mask priv_mask
*
public          9.34.22.0  255.255.255.0
monitor        9.34.22.0  255.255.255.0
NVTEST         9.34.22.0  255.255.255.0
password1      9.34.22.0  255.255.255.0
password2      9.0.0.0    255.0.0.0
password3      9.132.2.4  255.255.255.255
*-----*
*
* The following community names will turn tracing within the
* agent on/off when a packet arrives with such a community_name.
* This is achieved by specifying an appropriate priv_mask
*
*          /-----> bit 0 (trace on or off)
*          | /-----> bit 1 (trace SNMP responses)
*          || /-----> bit 2 (trace SNMP requests)
*          ||| /-----> bit 3 (trace external)
*          |||| /-----> bit 4 (trace internal)
*          ||||| /-----> bit 5 (trace DPI)
*          ||||| /-----> bit 6 (trace DPI internals)
*          ||||| /-----> bit 7 (reserved)
*          ||||| /-----> bit 31 (last bit)
*          vvvvvvvv          v
* Here, string xxxxxxxxxxxxxxxxxxxxxxxxxxxxs
* is used by the agent to grant access and turn tracing on/off.
* The string represents bits in a 32 bit integer.
* The last bit (if not set to s) means you have access.
* Bit zero turns tracing on (if set to s) or off (if set to x)
* Bits 1-7 specify which tracing is turned on or off.
*
debug_reply    9.34.22.0  255.255.255.0  ssxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_reply_off 9.34.22.0  255.255.255.0  xsxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_req      9.34.22.0  255.255.255.0  sxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_req_off  9.34.22.0  255.255.255.0  xxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_ext      9.34.22.0  255.255.255.0  sxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_ext_off  9.34.22.0  255.255.255.0  xxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_int      9.34.22.0  255.255.255.0  sxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_int_off  9.34.22.0  255.255.255.0  xxxsxxxxxxxxxxxxxxxxxxxxxxxxs
debug_dpi      9.34.22.0  255.255.255.0  sxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_dpi_int  9.34.22.0  255.255.255.0  sxxxxsxxxxxxxxxxxxxxxxxxxxxs
debug_all_dpi  9.34.22.0  255.255.255.0  sxxxxsxxxxxxxxxxxxxxxxxxxxxs
debug_dpi_off  9.34.22.0  255.255.255.0  xxxxsxxxxxxxxxxxxxxxxxxxxxs
debug_all      9.34.22.0  255.255.255.0  sssssssxxxxxxxxxxxxxxxxxxxxs
debug_all_off  9.34.22.0  255.255.255.0  xssssssxxxxxxxxxxxxxxxxxxxxs
debug_off      9.34.22.0  255.255.255.0  xssssssxxxxxxxxxxxxxxxxxxxxs
```

4. Comments and sequence numbers are not allowed in the PW SRC file.

The IP address of an incoming SNMP request is ANDed with the first network mask in the PW SRC file. If the result matches the corresponding network address, the community names are compared. If the names match the request is accepted, unless the privilege mask does not specify 's' in the last position. If

the network address or community names do not match, the search continues until all entries have been tested.

In the example in item “3” on page 443, if a request is received from IP address 9.34.22.122 with community name **password2**, the first four entries are skipped because the community names do not match. In the fifth entry, the community name does match, so IP address 9.34.22.122 is ANDed with the mask 255.0.0.0. The result is 9.0.0.0, which equals the specified network address for community name **password2**, so the request is accepted. For community name **password3**, only requests from host 9.132.2.4 will be accepted.

If the IP address ANDed with the network mask does not match a *desired_network*, or if the *community_names* do not match, an AUTHENTICATION_FAILURE TRAP is sent, provided that a destination entry exists in the SNMPTRAP DEST file.

It is considered good practice to allow requests only from the IP addresses on which the network management system operates. Allowing anyone to make requests may lead to the leak of sensitive information, and there is the potential that the system may be exploited for malicious purposes.

SNMP Daemon Installation Steps

This section describes the installation steps of the SNMP daemon.

SNMP Daemon

This section describes how to install the SNMP daemon (agent).

1. Ensure that the TCPIP DATA file and, optionally, the HOSTS ADDRINFO and HOSTS SITEINFO files are on a disk accessible to the SNMPD virtual machine.
2. Ensure that the SNMPTRAP DEST file (see “TRAP Destination file” on page 442), and the PW SRC file (see “PW SRC File” on page 442) are on a disk accessed by SNMPD.
3. Update the DTCPARMS file. For more information, see “Step 2: Update the DTCPARMS File for SNMPD and SNMPSUBA” on page 440.
4. Add SNMPD to the AUTOLOG, PORT, and OBEY statements in the PROFILE TCPIP file. For more information, see “Step 1: Update PROFILE TCPIP” on page 439.
5. Add the SYSCONTACT and SYSLOCATION statements in the PROFILE TCPIP file. For more information, see “Step 1: Update PROFILE TCPIP” on page 439.
6. If Bridge MIB variables are to be obtained for a virtual switch defined on the system, define an interface using the DEVICE, LINK, START, and HOME statement, including the VSWITCH keyword on the HOME statement. Follow the steps in “Setting up an SNMP Subagent” on page 444 to set up the SNMP Subagent.

Setting up an SNMP Subagent

This section describes the steps needed to set up an SNMP subagent that can be used to supply a specific set of MIB variables (such as BRIDGE-MIB variables for a z/VM Virtual Switch).

1. Create the SNMP subagent virtual machine. The default user ID for the SNMP subagent is SNMPSUBA. This virtual machine requires privilege class E in order to be able to issue DIAGNOSE X'26C'.
2. Update the DTCPARMS file as follows:
 - Make sure that the :Parms. entry for the SNMP agent (SNMPD) has specified the user ID of the subagent (SNMPSUBA) by using the **-s** parameter (this will cause the SNMP agent to autolog the subagent). For example,

```
:nick.SNMPD      :type.server :class.snmp
                  :owner.TCPMAINT
                  :parms.-s SNMPSUBA
```


- Make sure that the :Parms. entry for the SNMP subagent (SNMPSUBA) has specified the user ID of the SNMP agent (SNMPD) by using the **-u** parameter. For example,

```
:nick.SNMPSUBA :type.server :class.snmp_agent
:owner.TCPMAINT
:parms.-u SNMPD
```

3. Rename the MIB_EXIT SDATA file to MIB_EXIT DATA.
4. Use the sample exec, MIBX2DSC SAMPEXEC (shipped on the TCPMAINT 592 disk) to extract the variable names from the MIB_EXIT DATA file and append them to the MIB_DESC DATA file.
To do this, you must first rename the MIBX2DSC SAMPEXEC to MIBX2DSC EXEC and then make sure that both the MIB_EXIT DATA file and the MIB_DESC DATA files are accessible prior to the exec.
5. To obtain Bridge MIBs for a virtual switch, add the VSWITCH keyword and virtual switch name to an IP address on the HOME statement in the TCP/IP configuration file. Each virtual switch you want to monitor must be associated with a different IP address. For more information, see [Configuring an SNMP Subagent for a Virtual Switch in z/VM: Connectivity](#).

Adding User-defined MIBs to an SNMP Subagent

This section describes the steps needed to modify an SNMP subagent so it can return additional MIBs.

1. Customize the MIB_EXIT DATA file by adding the names of any exit routines that are to supply MIB data, followed by a list of the MIB variables that are to be supported by each such exit routine. The name of the exit routine must be listed in the MIB_EXIT DATA file using the **EXITNAME** statement (for example, EXITNAME SNMPMIBX TEXT). Beginning with the line after the EXITNAME statement, list each of the MIB variables that are supported by the named routine. A sample MIB_EXIT DATA file is supplied as file MIB_EXIT SDATA on the TCPMAINT 591 minidisk. To use the sample, copy it to TCPMAINT 198, modify it as needed, and rename it to MIB_EXIT DATA.
Note: The sample MIB_EXIT DATA file that is supplied contains sample entries for user-defined MIBs. These entries are coded to use the generic SNMPMIBX routine, which is a programming example. The sample SNMPMIBX ASSEMBLE code as shown requires the High Level Assembler (HLASM).
2. For any exit routines that are listed in the MIB_EXIT DATA file using the EXITNAME statement, the text deck for that routine must be on a disk that is accessible by the subagent virtual machine. If you intend to use the IBM-supplied SNMPBRGX exit routine (for returning BRIDGE-MIB variables), the subagent virtual machine should already have access to the SNMPBRGX TEXT file that is shipped on the TCPMAINT 591 disk. If you intend to use the generic SNMPMIBX sample exit routine (which is shipped on the TCPMAINT 591 as SNMPMIBX SAMPASM), copy and rename the source code, modify it as needed and compile it. (This will require the High Level Assembler.) The resulting text deck should be placed on a disk accessible by the SNMP subagent virtual machine (that is the TCPMAINT 198 disk).
3. Add all of the MIB variable names and short names to the MIB data file (MIB_DESC DATA) that was created in “Step 3: Create the MIB Data File” on page 441. To do this, you must first copy and rename the MIBX2DSC SAMPEXEC to MIBX2DSC EXEC and then make sure that both the MIB_EXIT DATA file and the MIB_DESC DATA file are accessible prior to executing the exec.

Configuring the SNMP Client

This section describes how to configure the SNMP client.

SNMP Client Overview

Figure 4 on page 446 illustrates the z/VM implementation of SNMP with NetView.

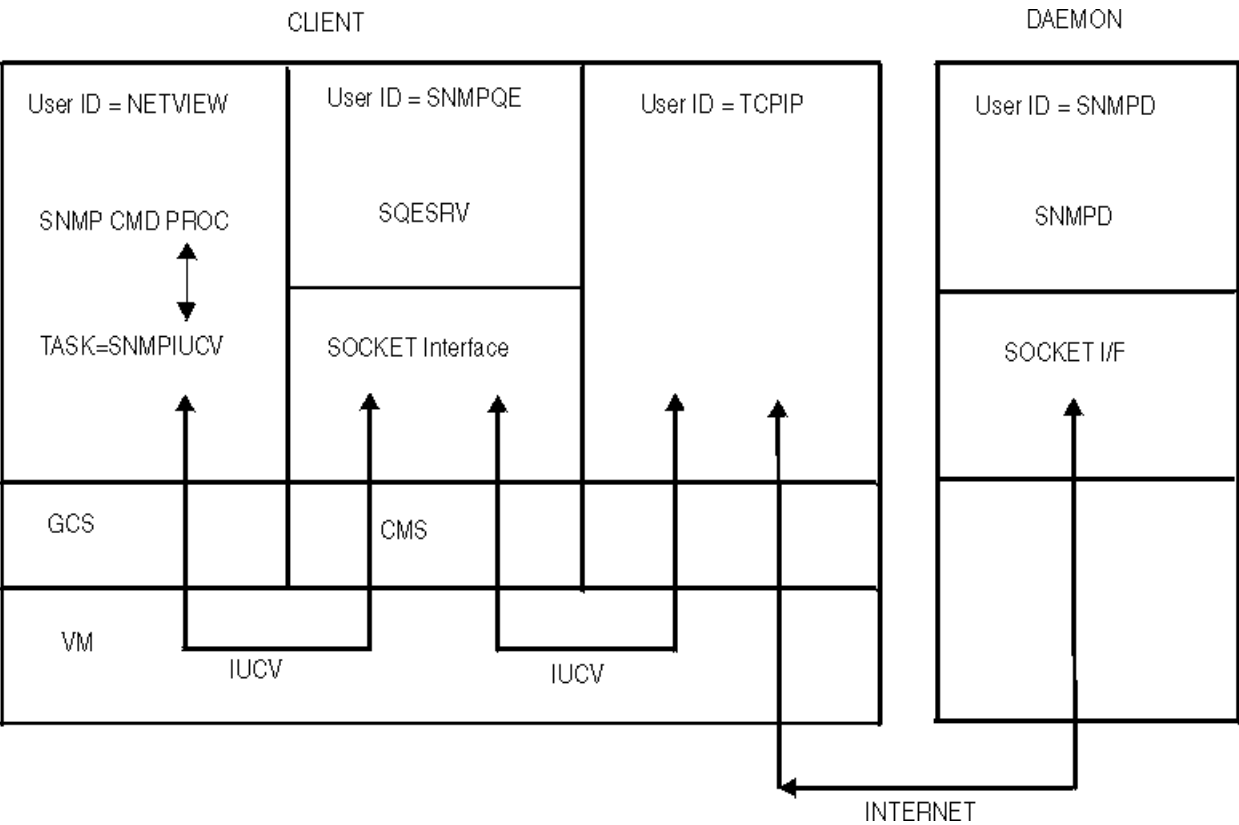


Figure 4. Overview of NetView SNMP Support

The following steps describe how the SNMP command is processed.

1. The NetView operator or CLIST issues an SNMP command.
2. The SNMP command is validated by the SNMP Command Processor.
3. The Command Processor passes the request to the SNMPIUCV task.
4. The SNMPIUCV task passes the request to the SNMP Query Engine*.
5. The SNMP Query Engine validates the request, converts the MIB variable to ASN.1 format if necessary, builds the SNMP request and sends it to the SNMP daemon.
6. The SNMP Query Engine receives a response from the SNMP daemon.
7. The SNMP Query Engine decodes the response and sends it to the NetView SNMPIUCV task.
8. The SNMPIUCV task sends the response as a multi-line message to the requesting operator or authorized receiver.

* The SNMP Query Engine is a separate z/VM user ID that runs the SQESERV MODULE. The SNMP Query Engine requires access to the MIB_DESC DATA file, which contains the MIB variable descriptions. A sample MIB data file called MIB_DESC_SDATA is supplied on the TCPMAINT 591 disk. Store your customized file on the TCPMAINT 198 disk as MIB_DESC DATA.

SNMP Client Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the DTCPARMS file for SNMPQE. 3. Create the MIB data file. 4. Configure the SNMP/NetView interface.

Step 1: Update PROFILE TCPIP

Include SNMPQE in the AUTOLOG statement to automatically start the SNMPQE virtual machine when TCPIP is invoked. Verify that the following statements have been added to PROFILE TCPIP.

```
AUTOLOG
  SNMPQE 0 ; SNMP Query Engine
```

SNMP requires that port UDP 162 be reserved for SNMP messages that report TRAPs. Verify that the following statements have been added to PROFILE TCPIP.

```
PORT
162 UDP SNMPD ; SNMP Client port for receipt of TRAPS
```

The SNMP Query Engine and agent use raw sockets for the DPI interface and the SNMP PING function. To allow the SNMP Query Engine (SNMPQE) to create RAW sockets, add SNMPQE to the OBEY list in the PROFILE TCPIP file. Verify that the following statements have been added to PROFILE TCPIP.

OBEY
SNMPOE

For more information on the OBEY list, see “OBEY Statement” on page 597.

Step 2: Update the DTCPARMS File for SNMPQE

When the SNMPQE server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the SNMPQE server are:

```
:Nick.SNMPQE
:parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

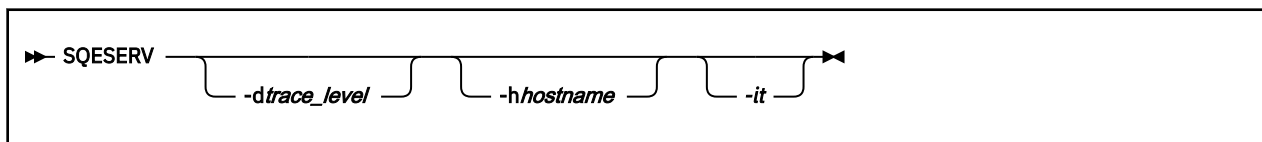
Note: You should modify the DTCPARMS file for the SNMPOE server if you:

- Activate and specify the level of tracing needed.
- Require a trace of IUCV communication to be done.
- Wish to have the SNMPOE server listen on a port other than port 162.

SQESERV Command Operands (:Parms. Parameters)

SNMPQE services are initiated using the SQESERV command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms, tag that is associated with an SNMPQE server definition. For more information about this command and its operands, see [“SQESERV Command Syntax”](#) on page 447.

SQESERV Command Syntax



Operands

-d trace level

Specifies the level of tracing to be run. Valid values for the trace level are:

- 0** No tracing (default)
 - 1** Display errors
 - 2** In addition to errors, also displays SNMP Query Engine protocol packets sent and received
 - 3** In addition to SNMPQE packets and errors, also displays the SNMP packets sent and received
- h *hostname***
Specifies the IP address to bind to, so that SQUERV accepts only connections through that IP address. This parameter is useful if multiple IP addresses exist for a single host and you want to restrict access from one side.
- it**
Specifies that a trace of IUCV communication is to be done. Used for debugging only the socket layer in a user's application. It can result in a very large amount of output.

Step 3: Create the MIB Data File

The Management Information Base (MIB) data file, MIB_DESC DATA, defines the short names for MIB variables. Short names are the character representation for the ASN.1 variable names. For example, sysUpTime is the short name for 1.3.6.1.2.1.1.3.0 (the MIB variable that stores the time since the MIB object was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP for VM ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP daemon. You can always use ASN.1 language to specify the variable names in an enterprise-specific tree (assuming that the agent supports them). You can use these short names to specify the MIB variables.

When you issue an SNMP GET, GETNEXT, or SET command, and specify the variable name in ASN.1 notation, the SNMP Query Engine uses that name and sends it in the SNMP packet to the agent. When you issue an SNMP GET, GETNEXT, or SET command, and specify the short name for the variable (for example, sysDescr), the SNMP Query Engine looks for that name in the MIB_DESC DATA file and uses the ASN.1 name specified in the file when it sends the SNMP packet to the agent.

The distributed MIB_DESC DATA file contains the text names as defined in RFC 1156. In addition to these, the following variables have been added:

- MIB-II variables from RFC 1158

The SNMPQE virtual machine must be able to access the MIB_DESC DATA file.

You can change the short names in the MIB_DESC DATA file to the equivalent in your national language. You can also leave the current names and add the equivalent names in your national language. However, the SNMP MIBVNAME function returns only the first entry found in the file that satisfies the search. In addition, all enterprise-specific variables used by hosts in your network should be added to this file.

Entries in the file do not need to be in a specific sequence. Each name starts on a new line. The following shows the line format.

```
short_name asn.1_name type time_to_live
```

Each variable on the line is separated by either one or more spaces or tabs. An asterisk (*) in column 1 indicates that the line is a comment line.

Figure 5 on page 449 is an example of an MIB_DESC DATA line with a sysDescr variable translated in Dutch and a few enterprise variables added (in this example, company ABC received 1.3.6.1.4.1.42 as the ASN.1 number for their enterprise). A sample MIB data file is supplied on the TCPMAINT 591 disk as MIB_DESC SDATA. Your customized file should be stored on the TCPMAINT 198 disk as file MIB_DESC DATA.

```

*-----*
* MIB Variable name | ASN.1 notation      | Type   | TTL | *
*-----*
sysDescr            1.3.6.1.2.1.1.1.      display 900
* Following is Dutch name for sysDescr
systeemBeschrijving 1.3.6.1.2.1.1.1.      display 900
...
other entries
...
* Following are enterprise specific variables for company ABC
ABCInfoPhone        1.3.6.1.4.1.42.1.1    display 900
ABCInfoAddress       1.3.6.1.4.1.42.1.2    display 900

```

Figure 5. Sample MIB_DESC DATA Line

The TTL field contains the number of seconds that a variable lives in the Query Engine's internal cache. If there are multiple requests for the same variable within the TTL period, the variable value is obtained from the cache, and unnecessary network traffic is avoided.

You can define multiple short names or text names for the same variable, as shown with the Dutch translation of the sysDescr variable. In this case, the SNMP Query Engine returns the first value in the table on an SNMP MIBVNAME request.

When the SNMP Query Engine receives a short name or text name in a GET, GETNEXT, or SET request, it compares the name against the entries in the MIB_DESC DATA file. This comparison is not case-sensitive. For example, a request for SYSDESCR, SysDescr, or sysDescr matches the sysDescr entry with an ASN.1 notation of 1.3.6.1.2.1.1.1..

When the SNMP Query Engine receives an SNMP response, it looks up the variable in the MIB_DESC DATA table Type field for information about translating the value into displayable characters. The information contained in the Type field is case-sensitive and must be specified in lowercase.

Step 4: Configure the SNMP/NetView Interface

This section describes how to configure the SNMP/NetView Interface.

SNMPIUCV

SNMPIUCV is the NetView optional task that handles IUCV communication with the SNMP Query Engine. You need to add the following TASK statement for SNMPIUCV to the DSIDMN NCCFLST file.

```
TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

This statement causes SNMPIUCV to start automatically when NetView is started. If you use INIT=N in the TASK statement for SNMPIUCV, a NetView operator can start the SNMPIUCV task by entering the following:

```
START TASK=SNMPIUCV
```

The SNMPIUCV task tries to connect through IUCV to the SNMP Query Engine. If this fails, it retries the connect as specified in the SNMPARMS NCCFLST file, see [“SNMPIUCV Initialization Parameters”](#) on page 450, the default is every 60 seconds.

SNMP Command Processor

SNMP is the command processor that allows NetView operators and CLISTs to issue SNMP commands. You should add the following statement to the DSICMD NCCFLST file.

```
SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=N
```

RES=Y to make the command resident in memory to improve performance.

After the SNMPIUCV task is started, you can issue the SNMP command. The SNMP command passes a request to the SNMPIUCV task to forward to SNMPQE. The return code represents a request number

that is associated with the request. The responses are returned asynchronously and contain this request number. The operator or CLIST must use the request number to correlate the response to the request.

SNMP Messages

The SNMP messages reside in the DSISNM nn NCCFLST files, which are shipped on the server code disk, TCPMAINT 591 (nn indicates the number of the message). The valid message files are DSISNM00 through DSISNM05, DSISNM10, DSISNM12, and DSISNM99. These files should reside on a disk that the NetView user ID can access.

SNMPIUCV Initialization Parameters

SNMPIUCV reads the SNMPARMS NCCFLST file at startup. This file contains the initialization parameters for SNMP and is shipped on the server code disk, TCPMAINT 591. You should place the SNMPARMS NCCFLST file on any disk that can be accessed by the NetView virtual machine.

The following example shows the parameters and the default values of the SNMPARMS NCCFLST file.

```
*
*      SNMPARMS NCCFLST
*
SNMPQE   SNMPQE   * Userid of SNMP Query Engine
SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
SNMPRCNT 2       * Retry count for sending SNMP requests
SNMPRITO 10      * Retry initial timeout (10ths of a second)
SNMPRETO 2       * Retry backoff exponent (1=linear, 2=exponential)
SNMPMMLL 80      * Line length for Multiline Messages 38/44
```

You can change the parameters in the SNMPARMS NCCFLST file. The following describes each of the keywords and parameters:

Parameter Description

SNMPQE *name*

Specifies the virtual machine of the SNMP Query Engine. The default is SNMPQE. If you change the name of the SNMP Query Engine virtual machine to something other than SNMPQE, you must also change this parameter to match. This value is case sensitive.

SNMPQERT *seconds*

Specifies the retry timer (seconds) for IUCV CONNECT. When SNMPIUCV is started, it tries to connect to the SNMP Query Engine. If the connection fails or breaks, SNMPIUCV retries a connect every n seconds, as specified by this parameter. The valid range of values is 0 to 9999. The default is 60 seconds.

SNMPRCNT *number*

Indicates the retry count for sending SNMP requests. This is the number of times the SNMP Query Engine resends an SNMP PDU when no response was received. If no response was received after all retries have been exhausted, the SNMP Query Engine returns a no response error for the SNMP request. The valid range of values is 0 to 255. The default is 2.

If the request being sent by the SNMP Query Engine contains an invalid community name, no response is received. This causes the SNMP Query Engine to resend the request until the retry count is exhausted. The agent generates multiple authenticationFailure traps, one for the initial request and one for each retry.

SNMPRITO *tenths_seconds*

Specifies the time-out value for the request in tenths of a second. After sending an SNMP request to an agent, the SNMP Query Engine waits the specified time for a reply. If no reply is received within the specified time limit, the SNMP Query Engine resends the request the number of times specified by SNMPRCNT. If no replies have been received after all retries have been exhausted, the SNMP Query Engine returns a NO RESPONSE error to NetView. The valid range of values is 0 to 255. The default is 10 tenths of a second.

SNMPRETO *exp*

Indicates the retry back-off exponent. Specifies whether the time-out value between retries of an SNMP request is calculated linearly or exponentially. The valid values are 1 (linear) or 2 (exponential). The default is 2.

For example, if the retry time-out was 1 second, SNMPRETO of 1 causes a new retry to be sent at constant one second intervals until all retries have been sent. SNMPRETO of 2 causes the first retry to be sent after one second, the second retry 2 seconds later, the third retry 4 seconds later, and so on until all retries have been sent.

SNMPMMLL *length*

Indicates the line length for multiline messages 38 through 44. The maximum length is 255. A value of 80 allows the complete text to appear on an 80 character-wide screen. The default length is 80 characters.

SNMP Client Installation Steps

SNMP requires NetView as the client of the local host.

SNMP Command Processor and SNMPIUCV on NetView

This section describes how to install the SNMP Command Processor and SNMPIUCV.

1. If necessary, move the SNMPLIB LOADLIB to the NetView 191 disk or any other disk to which NetView has access, from the server code disk, TCPMAINT 591, where the file was shipped.
2. Add this load library to the GLOBAL LOADLIB statement in the NetView start-up procedure. This file has a file type of GCS. For example, the following is a typical GLOBAL command.

```
GLOBAL LOADLIB STATMON NLDM NPDA PROPMX USER
```

Now change it to:

```
GLOBAL LOADLIB STATMON NLDM NPDA PROPMX SNMPLIB USER
```

As an alternative, you can add the SNMP and SNMPIUCV load modules to the USER LOADLIB.

3. Add a TASK statement to the DSIDMN NCCFLST file.

```
TASK MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

If you code INIT=N on the TASK statement for SNMPIUCV, the NetView operator must start the task manually with the following command.

```
START TASK=SNMPIUCV
```

4. Add the following statement to the DSICMD NCCFLST file.

```
SNMP CMDMDL MOD=SNMP,ECHO=Y,TYPE=R,RES=N
```

If you issue many SNMP commands, you can use RES=Y to make the command processor resident.

5. If necessary, put all the DSISNM nn NCCFLST files on the NetView 191 disk (or a disk accessible to NetView) from the server code disk, TCPMAINT 591, where the files were shipped. These are the externalized messages.
6. If necessary, put the SNMPARMS NCCFLST file on the NetView 191 disk or any other disk to which NetView has access, from the server code disk, TCPMAINT 591, where the file was shipped. These are the SNMP startup parameters. You can change these parameters if the default values are not acceptable. For more information, see [“SNMPIUCV Initialization Parameters” on page 450](#).

Chapter 15. Configuring the SSL Server

The Secure Socket Layer (SSL) server provides the processing capability that allows secure (encrypted) communication between two TCP/IP connection participants (one of which is a server or client application on the local z/VM host). Such communication may be secured by a static SSL connection (one that is secured when the connection is initially established, and remains as such for the duration of the connection) or through Dynamic SSL/Transport Layer Security (TLS), which allows a client or server application to control the acceptance and establishment of connections that are encrypted using SSL.

For z/VM, implicit (*static*) SSL connections are supported only between a remote client and an application (or, protocol) server that resides on the local z/VM host. The application server must be listening on a port identified as SECURE by your installation, and the remote client must support the SSL protocol according to RFC 2246. Explicit (*dynamic*) SSL/TLS connections are supported for any z/VM Pascal or Assembler client or server application that makes use of the set of application programming interfaces (APIs) provided for this purpose. For more information about the APIs provided for using Dynamic SSL/TLS, see *z/VM: TCP/IP Programmer's Reference*.

For static SSL connections, no changes to a z/VM application server are necessary to participate in SSL. The application server does not perform any data encryption or decryption; this is handled by the z/VM SSL server.

Dynamic SSL/TLS connections are supported by the following z/VM TCP/IP application servers and clients, which have been updated to accommodate this support:

- TCP/IP server
- SSL server
- FTP server
- FTP client
- Telnet server (Internal to the TCP/IP server)
- Telnet client
- SMTP server

Note: The LDAP server makes use of SSL/TLS services that are separate from those provided by the z/VM SSL server. Thus, you do not protect the LDAP server ports in the same manner as that described here, for other servers such as the FTP server.

Under SSL protocol, the application server is always authenticated. To participate in an SSL session, an application server must provide a certificate to prove its identity. Server certificates are issued by Certifying Authorities (CAs), each of which establishes its own identity by providing a CA certificate. Server certificates and CA certificates are stored in a certificate database (also referred to as a *key database*) that is accessible to the SSL server. The key database resides in the z/VM Byte File System (BFS) and is managed independent of the SSL server, through the use of a utility program, **gskkyman**.

To configure the SSL server, you must perform the following steps:

SSL Server Configuration Steps
1. Determine the SSL Server Configuration For Your Installation 2. Update the TCP/IP Server Configuration File (PROFILE TCPIP) 3. Update the DTCPARMS File for the TCP/IP Server. 4. Update the DTCPARMS File for the SSL DCSS Management Agent Server. 5. Update the DTCPARMS File for the SSL Server Pool 6. Set up the certificate database. 7. Implement Customization for Protected Communications a. Designate the Secure Ports (Static SSL Connections) b. Configure TLS Services (Dynamic SSL/TLS Connections)

Dynamic Server Operation: The SSL server provides an SSLADMIN command interface that allows you to perform certificate database administration and server administration tasks. See [“Dynamic Server Operation”](#) on page 485.

Overview of an SSL Session

The general processing steps associated with an SSL connection are described here. Note that these steps are more descriptive of a static SSL connection for a z/VM application server than for a TLS connection. For TLS, similar processing is performed, although the SSL server does not participate when the initial connection is established, since the connection is a clear connection (one that is not secured). The interaction with the SSL server occurs once a Handshake request is initiated by the client or the application server (with TLS, data encryption is often controlled at the direction of the client, through subsequent renegotiations for the duration of the connection). SSL and TCP/IP server interactions similar to those described would also be applicable to z/VM client programs that use TLS, when connections are established with a remote server.

An SSL session consists of the following general processing steps:

1. Connect

A remote client sends a connection request for an application server. If the application server is listening on a secure port, the TCP/IP (stack) server sends this request to the SSL server. The TCP/IP server also sends a label that identifies the certificate to be used for the secure connection, as well as the socket addresses of the client and the application server. The SSL server accepts the connection from the client and sends a connection request to the application server.

The SSL session is maintained as two separate connections: the connection from the remote client to the SSL server, and the connection from the SSL server to the application server. The intervention of the SSL server is transparent to the client and the application server; to them, it seems that they are communicating directly with each other.

2. Handshake

After its connection request is accepted, the client initiates a handshake protocol to produce the cryptographic parameters for the session. The SSL server (representing the application server) responds to the handshake and sends the application server's certificate to the client. The SSL server might then ask the client for its client certificate and then attempt to authenticate it.

Clients that make use of SSL services generally have the certificates associated with the Well Known CAs in their certificate databases. The client compares the "signature" on the application server's certificate with the appropriate CA certificates to verify the authenticity of the server.

When the SSL server is requesting client certificates, the CA that signed the client's certificate must be in the SSL server's certificate database. The SSL server can then verify authenticity by comparing the "signature" of the client's certificate with the appropriate CA certificate.

The client and the SSL server then agree on a protocol version, select cryptographic algorithms (known as cipher suites), and use asymmetric (public-key) encryption techniques to generate shared secrets. From the shared secrets, the SSL server and the client generate the symmetric (private) keys to be used for the encryption and decryption of data sent on the connection.

3. Data transmission

When the handshake completes, the client sends encrypted data over the network. The SSL server receives the encrypted data from the client, decrypts it, and sends it to the application server. The application server responds by sending unencrypted data to the SSL server. The SSL server receives the unencrypted data from the application server, encrypts it, and sends it to the client.

4. Close

When a close request is received from either the client or the application server, the SSL server sends a close request to the other party and cleans up the connection.

Understanding Certification Validation

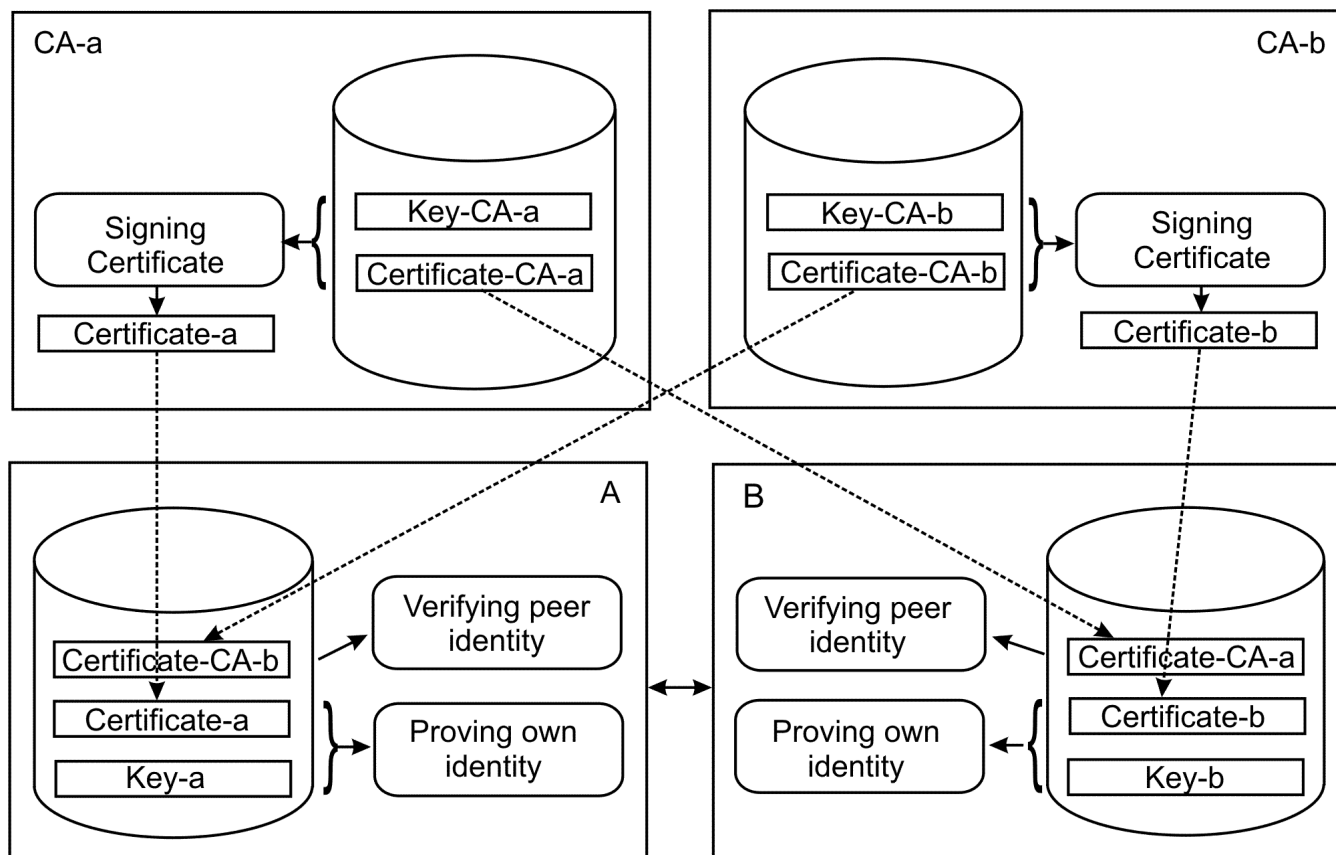
One key feature of the SSL protocol is the ability to validate the identity of a remote participant of a secure session. In general, either of the two participants (a client and a server) may or may not request validation of its peer's identity. For some client applications, including those provided with z/VM, server identity validation is mandatory, which cannot be overridden.

Given that one or both participants in a session may not have any prior knowledge about its peer, each can rely on a third party (conventionally referred to as a Certification Authority, or CA) that is known to (and, trusted by) both communicating parties, to certify the identity of a peer.

Technically, a participant that needs to prove its identity to a peer must have a Private Key and a corresponding Certificate at its disposal. The certificate is the Public Key that corresponds to the Private Key, signed by the key of a Certification Authority. The participant that requests identity validation of its peer needs the Certificate (the signed Public Key) of the Certification Authority that signed the peer's Certificate.

Consider first, an example in which participants A and B both request identity validation of their respective opposite.

Participant A has its Key-a and Certificate-a, the latter signed by the Certification Authority CA-a. Participant B has its Key-b and Certificate-b (which is signed by a Certification Authority different from CA-a, that being Certification Authority CA-b). For this mutual validation to work, participant A must also have access to the Certificate of CA-b. Likewise, participant B must have access to the Certificate of CA-a. In other words, each participant needs access to its own Key and its own Certificate, in addition to the Certificate of the CA that signed the Certificate used by its peer.



Now consider a different scenario, in which participant A requests validation of participant B's identity, but B has no interest in validating the identity of participant A. Participant B must have its Key and Certificate, whereas A must possess the Certificate of CA-b, and does not need anything else. (For technical reasons, if A is a server, it still needs a Key and Certificate, but the signer of the Certificate is of no importance).

Of course, it is possible that both participants might attempt to validate each other when their Certificates have been signed by the same CA. In this case, each participant must have access to the Certificate of this common CA to validate its peer's identity.

Certification Authorities and Self-Signed Certificates

There exist a number of "universally recognized" Certification Authorities that will sign others' Certificates for a fee, under certain conditions. Such conditions usually require that the requestor of the certificate provide proof of identity, that can be corroborated by the certifying authority, using written or electronic means, or a combination of these.

For identity validation of the parties operating within a single company or organization, an internal Certification Authority can be established by that company or organization. A Certification Authority of this type can produce certificates that are trusted by all interested parties and distributed among them for use in securing localized communications.

Self-signed certificates are a special case of the above. When a Certificate is signed by itself, it has two roles:

- First, together with the Key, it provides the proof of identity of a communication participant.
- Secondly, it functions as the Certificate of the CA that has signed the Certificate.

Note: The combination of Certificate and Key is required to provide a "proof of identity", whereas the Certificate without the Key is sufficient (and appropriate) for serving the role of a "CA Certificate".

The use of self-signed certificates is typically not recommended for production environments. Such certificates should be used only to facilitate environment testing prior to production use. Self-signed

certificates do not imply the level of security or authenticity of the certificate signed by a certificate authority.

The use of self-signed certificates can be illustrated in these two examples:

1. Assume Client A connects to Server B, and neither participant has an interest in validating its peer's identity. In this case, Server B can make use of a self-signed Certificate with Key, and does not need anything else. Client A does not need any keys or certificates -- it needs only to acknowledge the use of Server B's certificate and key for securing the communication. (As previously noted, this scenario is not possible when a z/VM client application is used).
2. Now, assume instead that Client A connects to Server B, and requests validation of Server B's identity. Server B however, has no interest in validating the identity of Client A. In this case, it is possible for Server B to use a self-signed Certificate with Key. However, this same Certificate also must be imported (without its corresponding Key) to the certificate store or database used by Client A. Here, the same Certificate serves the role of "proof of identity" on the Server B system (together with the Key), and the role of "Certificate of the CA" on the Client A system.

SSL/TLS Partner Certificate Revocation Checking

The SSL server can be configured to obtain revocation information about a partner's certificate during the authentication step of connection establishment to allow or disallow SSL/TLS connection establishment. Any combination of the following revocation sources can be used:

- One or more dedicated Online Certificate Status Protocol (OCSP) responders specified in the Authority Information Access (AIA) extension of the partner's certificate
- An HTTP server specified in the CRL (Certificate Revocation List) Distribution Point (CDP) extension of the partner's certificate.

Enabling OCSP Support

OCSP revocation information can be obtained through OCSP responders identified within the partner's certificate AIA extension (URI value) or through a dedicated OCSP responder.

To enable the use of the certificate AIA extension, when it is present in the certificate that is being validated, the OCSPENABLE value must be set to ON. The OCSPENABLE setting is controlled by operands specified for the DTCPARMS file :OCSPParms. tag. For more information, see [“The :OCSPParms. Tag” on page 459](#). In order to use the AIA extension, the extension must contain at least one identifying OCSP responder entry. An OCSP responder entry contains an access method of OCSP and a URI access location.

When processing the possible values in the AIA extension, an attempt to contact each OCSP responder is tried and processing stops with the first responder that is able to be contacted successfully. The OCSP response from that OCSP responder is used to determine the revocation status.

To enable the use of a dedicated OCSP responder, OCSPURL must be set to the HTTP URL and port of the OCSP responder. Only one dedicated responder can be specified.

If OCSPENABLE is set ON and the dedicated OCSP responder is specified (OCSPURL), the dedicated OCSP responder and the responders identified in the AIA extension can be used to obtain revocation status. By default, the dedicated OCSP responder is used first during validation of a certificate. The default order can be changed through the OCSPRIORITY setting. To have the dedicated OCSP responder (OCSPURL) used after the AIA extension, OCSPRIORITY must be set to OFF.

In all cases, after an OCSP responder has returned a response, the response is used to determine the revocation state of the certificate being validated.

If the dedicated OCSP responder (OCSPURL) requires the OCSP request to be signed digitally or your security policy requires signing, signing is enabled by specifying the certificate to be used for signing and optionally the signature algorithm. The signing certificate is identified by setting OCSPSIGLABEL to the label of the certificate that can be found in the certificate database currently in use. The optional signature algorithm is specified through OCSPSIGALG and defaults to RSA with SHA-256 (0401).

OCSPSIGLABEL and OCSPSIGALG are only used to sign requests to the URL specified in OCSPURL or a dedicated OCSP responder.

All OCSP responses are signed digitally by the OCSP responder using a CA certificate. The response signature must be verified using the public key of that certificate. The OCSP signing certificate or certificate chain must be found in the certificate database the SSL server is configured to use to succeed.

The OCSP signing certificate will follow the OCSP policy that is currently in force, which means its revocation status will also be checked against a dedicated OCSP responder, a dedicated OCSP responder based on the configuration values in place, or both.

You can tailor the method used to contact an OCSP responder by using OCSPVIAGET to indicate whether an HTTP GET or HTTP POST will be used. By default, the HTTP POST method is used. The HTTP GET method allows for the enablement of HTTP caching on the OCSP responder and can only be used when the OCSP request is less than 255 bytes.

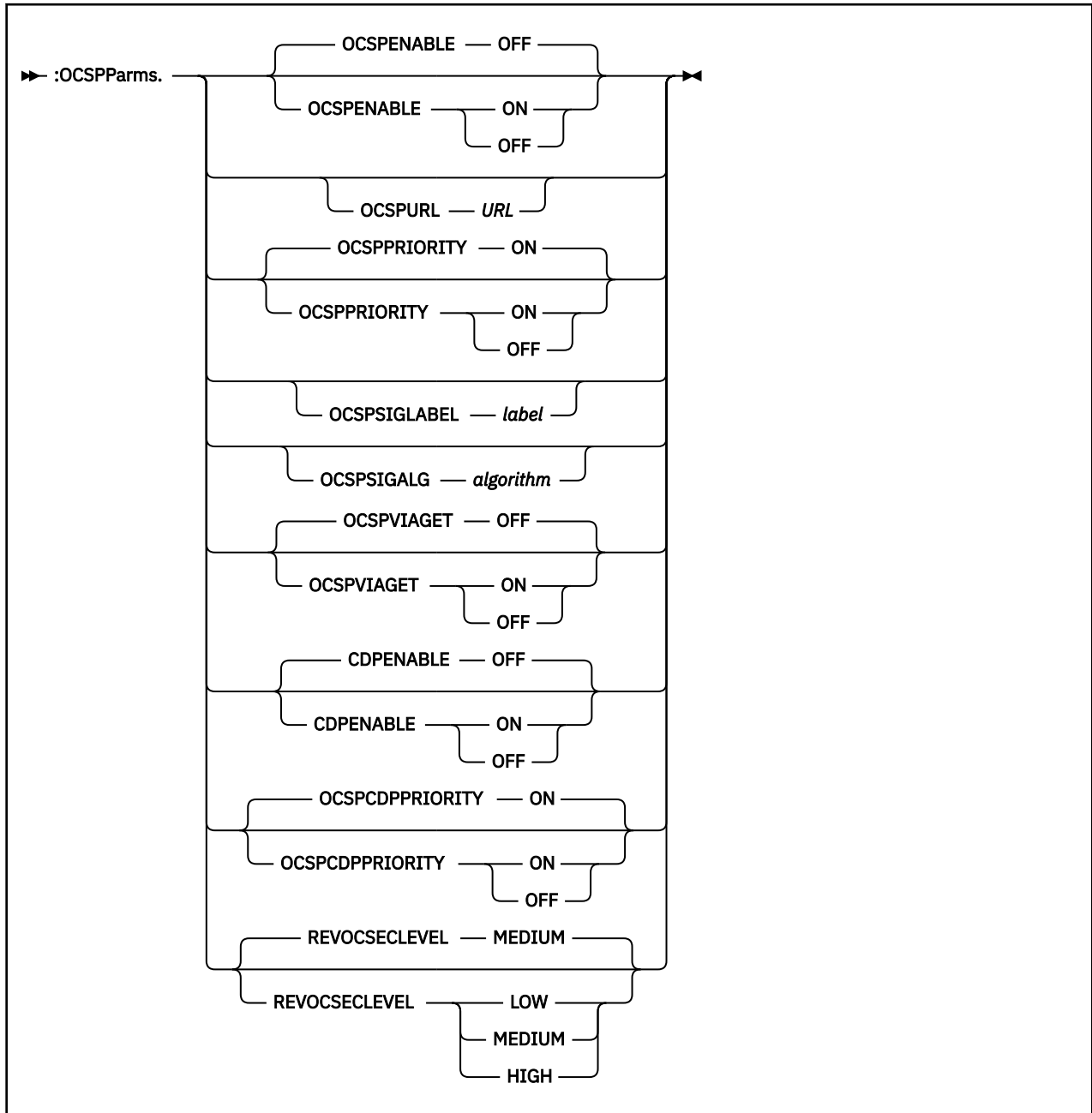
Enabling HTTP CDP Support

Certificate revocation can be obtained through HTTP servers identified within a partner certificate's CDP extension. The HTTP server provides the revocation information in the form of a CRL.

To enable the use of the certificate's CDP extension when the extension is present in the certificate that is being validated and for retrieving CRLs from an HTTP server, the CDPENABLE setting must be set to ON. The CDPENABLE setting is controlled by operands specified for the DTCPARMS file :OCSPParms. tag. For more information, see [“The :OCSPParms. Tag” on page 459](#). In order to use the HTTP CDP extension, the extension must contain at least one identifying HTTP entry.

When processing the possible values in the CDP extension, an attempt to contact each HTTP server is tried and processing stops with the first server that is able to be contacted successfully. The HTTP response contains a CRL that is used to determine the revocation status of the certificate.

The :OCSPParams. Tag



Purpose

Use the `:OCSPParams.` tag to configure OCSP- and CDP-related information, if you are using OCSP or CDP certificate revocation checking or both.

Operands

OCSPENABLE

Enables the use of the AIA extension to locate one or more URIs of one or more OCSP responders. A minimum of one responder must be present in the extension. Processing stops at the first responder that can be contacted and that information is used to determine revocation status.

ON

The use of the AIA extension is enabled.

OFF

The use of the AIA extension is not enabled. This is the default.

OCSPPURL URL

Specifies the HTTP URL of a dedicated HTTP responder. Only one can be present. There is no default value.

OCSPPRIORITY *priority*

If both OCSPPENABLE and OCSPPURL are set, this determines the order of the responders that are checked. The default is to use the dedicated responder first, then proceed to the responders in the extension.

ON

The dedicated responder specified in OCSPPURL is used first. This is the default.

OFF

The URIs of the OCSPP responder found in the AIA extension are used first.

OCSPPSIGLABEL *label*

Specifies the label of the certificate in the certificate database that are used to sign an OCSPP request if a certificate responder requires a signed request. The algorithm that will be used can be set in OCSPPSIGALG if the responder requires an algorithm that is not the default algorithm. There is no default value.

Notes:

1. *label* can be 1 to 8 characters in length.
2. *label* should be specified using uppercase characters. (If needed, *label* will be converted to uppercase.)
3. OCSPPSIGLABEL is used only to sign requests to the URL specified by OCSPPURL or to a dedicated OCSPP responder.

OCSPPSIGALG *algorithm*

Specifies the signature algorithm that will be used if OCSPPSIGLABEL contains a label. The default is to use RSA with SHA-256 (0401 or SHA256_RSA). OCSPPSIGALG is used only to sign requests to the URL specified by OCSPPURL or to a dedicated OCSPP responder.

algorithm can be any of the following values:

```
0201 or SHA1_RSA      : SHA-1 with RSA
0202 or SHA1_DSA      : SHA-1 with DSA
0203 or SHA1_ECDSA     : SHA-1 with ECDSA
0301 or SHA224_RSA     : SHA-224 with RSA
0302 or SHA224_DSA     : SHA-224 with DSA
0303 or SHA224_ECDSA   : SHA-224 with ECDSA
0401 or SHA256_RSA     : SHA-256 with RSA
0402 or SHA256_DSA     : SHA-256 with DSA
0403 or SHA256_ECDSA   : SHA-256 with ECDSA
0501 or SHA384_RSA     : SHA-384 WITH RSA
0503 or SHA384_ECDSA   : SHA-384 WITH ECDSA
0601 or SHA512_RSA     : SHA-512 WITH RSA
0603 or SHA512_ECDSA   : SHA-512 WITH ECDSA
```

OCSPPVIAGET

Specifies whether an HTTP POST or GET should be used when contacting an OCSPP responder. The default is to use an HTTP POST (OFF).

ON

An HTTP GET is used when contacting an OCSPP responder.

OFF

An HTTP POST is used when contacting an OCSPP responder. This is the default.

CDPENABLE

Enables checking for the CDP extension in the certificate for the URL of an HTTP server to get revocation information. At least one HTTP URL must be present. If more than one is present, processing stops at the first URL that responds.

ON

If a CDP extension is found, it is used to obtain revocation information.

OFF

The certificate extension is not used if it is present. This is the default.

OCSPCDPPRIORITY

Specifies the order that the AIA and CDP extensions are checked if both CDP and OCSP processing are being used.

ON

OCSP responders are checked first, then any HTTP servers found in the CDP extension are used for revocation information.

OFF

Any HTTP servers found in the CDP extension are used first for revocation information, followed by any OCSP responders.

REVOCSECLEVEL level

Sets the security level when contacting OCSP responders or an HTTP server specified in an CDP extension.

LOW

Certificate validation does not fail if the OCSP or HTTP server in a CDP extension cannot be contacted.

MEDIUM

Certificate validation fails if revocation information cannot be obtained from any of the OCSP responders or an HTTP server specified in an CDP extension. If an OCSP responder or HTTP server is contacted, it must return a valid OCSP response or CRL.

HIGH

In addition to the MEDIUM requirements, if either or both CDP and OCSP are enabled via certificate extensions, certificate information must be provided in the client certificate or the revocation will fail. Certificate validation will fail if information cannot be obtained from any of the specified sources.

Usage Notes

1. TCP/IP applications that use the CLIENTCERTCHECK value of PREFERRED will conflict with OCSP and CDP revocation checking. If OCSP or CDP reports the certificate as being revoked, the connection will not be allowed, even though the value of CLIENTCERTCHECK is PREFERRED. A warning message will appear in the SSL console.
2. For CDP only, the first CDP extension is used if multiple CDP extensions are in the client certificate.
3. For OCSP, the first responder that can be contacted and responds is used for revocation status.
4. Only one dedicated OCSP responder can be specified.
5. If multiple revocation sources are enabled (OCSP and HTTP CDP), there is an order of precedence that is used when checking for certificate revocation information. Use the OCSPRIORITY setting to specify the order that the dedicated OCSP responder and OCSP responders in the AIA extension are checked for certificate revocation information.
 - To have the dedicated OCSP responder checked first, set OCSPRIORITY to ON. This is the default.
 - To have the AIA extension checked first, set OCSPRIORITY to OFF.
6. The OCSPCDPPRIORITY setting specifies the order that the AIA and CDP extensions are checked for certificate revocation information. To have the OCSP responders (dedicated OCSP responder or OCSP responders in the AIA extension) checked before the HTTP servers specified in the CDP extension, set OCSPCDPPRIORITY to ON. To have the HTTP servers specified in the CDP extension checked before the OCSP responders, set OCSPCDPPRIORITY to OFF. By default, this setting is set to ON.
7. In general, the first OCSP responder or HTTP server that responds with revocation information is used, even though more than one could be contacted and have revocation information.

8. OCSPSIGNAL and OCSPSIGNALG are used only to sign requests to the URL specified by OCSPURL or to a dedicated OCSP responder.
9. The OCSP signing certificate will follow the OCSP policy that is currently in force, which means its revocation status will also be checked against a dedicated OCSP responder, a dedicated OCSP responder based on the configuration values in place, or both.

Step 1: Determine the SSL Server Configuration For Your Installation

Secure communications support can be provided via one of the following SSL configurations:

- a single SSL server (a server pool comprised of only one server)
- a true, multiple-server "pool," for which several SSL servers are employed.

Single Versus Multiple SSL Server Configurations

The SSL server configuration required for a given installation depends largely on current (and, expected) capacity requirements for the provision and use of secure communications support. For some installations, use of a single SSL pool server might suffice, whereas for others, the use of an SSL server pool of multiple servers would be more appropriate.

Some general guidelines — based on IBM's performance analysis of an environment for which a default SSL server pool is defined, comprised of five servers with each server configured to support a maximum of 600 concurrent secure sessions — are provided here:

- To support a maximum of 100 concurrent secure connections, a single SSL server should suffice. For such a configuration, a minimal server pool (comprised of just one server), should be used.
- To support 100 to 600 concurrent connections, a server pool of five servers, with each defined to support a maximum of 120 sessions, should be considered. The lowered maximum number connections per server for this configuration likely will reduce CPU consumption on a per-server basis.
- To support more than 600 concurrent connections, use of the IBM defaults for the size of the server pool and the SSLIMITS statement should be considered.

Consult [Chapter 16, "Configuring the TCP/IP Server,"](#) on page 507 for more information about the ["SSLIMITS Statement"](#) on page 616 and ["SSLSERVERID Statement"](#) on page 617.

The sections that follow provide more information and considerations that can help determine which configuration might best be suited to your z/VM installation.

SSL Server Pool (Multiple Server) Considerations

The use of a server pool allows additional SSL servers to be deployed for supporting an increased number of secure connections, without the need to stop, and then restart, the TCP/IP server. This can be accomplished by implementing a CP directory change that increases the size of the server pool, and then using the OBEYFILE (or, NETSTAT OBEY) command to update the configuration established by the SSLIMITS statement.

When multiple SSL pool servers are defined, the TCP/IP server attempts to utilize a given server to its full capacity before an additional peer server is brought into use. Thus, secure connection requests are directed to the first "active" pool server, until that server has reached the maximum number of per-server connections defined for the server pool. Only when the server has reached this maximum, will connections be directed to the next (active) server. Should the number of active servers be insufficient to accommodate new connection requests, the TCP/IP server then attempts to initialize and employ any pool servers that previously have been determined to be "eligible" for use.

To ensure that secure connections are handled in a consistent manner by the various servers that comprise an SSL server pool, these servers must be uniformly configured, with respect to DTCPARMS attributes and server command parameters. As such, a specific pool server cannot be initialized (or, stopped and then restarted) with a configuration that differs from the remainder of the pool.

When changes to DTCPARMS configuration parameters are required, all members of the SSL server pool first must be stopped, after which the necessary changes can be implemented, with the entire server pool then restarted. Such changes cannot be implemented without interrupting secure communications services.

Note: This requirement does not affect the ability to make changes to the key database without such an interruption to services. SSL servers that currently are in operation can be made aware of such changes through use of the SSLADMMDIN REFRESH command.

The use of an SSL server pool requires the various pool servers to be enrolled in a CMS Shared File System (SFS) file pool, and that certain SFS directories be defined for use by these servers. For more information, see [“The SSL Pool Server Virtual Machine” on page 463](#), which can be used to implement these SFS requirements.

Single SSL Server Considerations

A single SSL server (a server pool comprised of just one server) can accommodate at most 1000 concurrent secure connections, which establishes this same limit for the overall number of such connections that can be handled by the associated TCP/IP stack server. Should the need arise to accommodate more secure connections, the single server pool can be redefined via a CP directory change that increases the size of the server pool, and then using the OBEYFILE (or, NETSTAT OBEY) command to update the configuration established by the SSLLIMITS statement.

Note: Multiple, separately-defined SSL servers cannot be used with a given TCP/IP stack server, in the same manner as can an SSL server pool.

The SSL Pool Server Virtual Machine

An SSL "pool" server (user ID SSLnnnnn) generally is one of several SSL servers that are defined for use with a given TCP/IP stack, to allow for more efficient handling of a large number of secure connections than can be accomplished by a single SSL server.

As noted in [“SSL Server Pool \(Multiple Server\) Considerations” on page 462](#), all of the servers that comprise an SSL server pool must be uniformly configured.

In addition, all pool servers must be enrolled in a CMS Shared File System (SFS) file pool, which facilitates the use of a common SFS directory instead of a 191 minidisk, as is used for other TCP/IP servers.

Note:

- The servers that comprise an SSL server pool cannot be shared among differing TCP/IP stack servers on the same system — a separate server pool must be defined for each such TCP/IP server.
- The SSLPOOL command can be used to create directory entry samples (and other configuration data) that pertain to the use of a specific SSL pool prefix and TCP/IP stack server. In addition, this utility can be used to enroll the various pool servers in a designated SFS file pool and further prepare the operating environment for these servers. For more information, see [“SSLPOOL Command” on page 502](#).

The SSL DCSS Management Agent Virtual Machine

The SSL DCSS Management Agent server (SSLDCSSM) creates and then serves as the owner of a Discontiguous Saved Segment (DCSS), which is used by an SSL server pool to maintain session cache information. The content of this cache is shared and maintained by all member servers.

Because of the role fulfilled by the SSL DCSS Management Agent server, it must be initialized prior to any of the SSL servers on whose behalf the server operates. Such initialization is accomplished automatically, when the TCP/IP stack server with which the agent server is associated, is initialized.

Cryptographic Mode Requirements and Configuration

Depending upon local security policy, implementing one or more modes of cryptographic operation may be a requirement for SSL server configuration. By default, the SSL server operates with no restrictions on key lengths or cipher suites. TLS 1.1 and TLS 1.2 are enabled by default, with other versions of SSL/TLS

disabled. While integrity checking of the certificate database is done during initialization, no other Known Answer Tests are performed.

z/VM currently provides two "mode" settings: FIPS-140-2 and NIST-800-131A. Either or both of these settings can be configured for the SSL server or an SSL pool; in cases where security settings may conflict, the SSL server will use the maximal common subset of functionality allowed for these settings.

MODE FIPS-140-2

The z/VM SSL server can be configured to operate in a mode compliant with Federal Information Processing Standard (FIPS) 140-2. This mode of operation changes SSL server operation in the following manner:

- Only TLS connections are permitted; SSL v3 and SSL v2 are automatically disabled.
- Key exchange lengths for RSA, DSA, and Diffie-Hellman must meet a minimum length of 1024.
- A FIPS-compliant certificate database is required. For more information on creating a FIPS-compliant certificate database, consult [SSL Certificate/Key Management and SSL Tracing Information in z/VM: TCP/IP User's Guide](#).
- Certain cipher suites will automatically be exempted from use. Consult [Table 39 on page 475](#) for information about which cipher suites are considered FIPS-compliant.
- Integrity checking and Known Answer Tests will be performed to validate the cryptographic environment.

MODE FIPS-140-2 can be enabled through one of two methods. The MODE operand of the "VMSSL Command Syntax" on page 470, described later in this chapter, allows the SSL server to determine FIPS mode as part of its initialization process. Depending upon security policy requirements (and per NIST Implementation Guidance), your installation may determine that it is necessary to configure FIPS 140-2 compliance prior to initialization.

The second mechanism for enabling FIPS-140-2 compliance is through an environment variable, GSK_DEFAULT_FIPS_STATE. When this variable is assigned the value "GSK_FIPS_STATE_ON", then the System SSL library will automatically enter FIPS-compliant mode as described above. The z/VM SSL server will detect this state and automatically configure itself as 'FIPS enabled.'

It is recommended that use of the GSK_DEFAULT_FIPS_STATE environment variable be done through a common TCP/IP exit (such as TCPRUNXT) so that a pool of SSL servers can maintain their common security policy.

In order to bring your TLS server into alignment with the configuration used for the z/VM 7.2 FIPS 140-2 validation, directory entry changes are required. It is strongly recommended that the PROFILE entry used by the TLS server pool (TCPSSLU, by default) be updated to IPL ZCMS instead of IPLing CMS. Additionally, the MACHINE mode should be set to Z. For example, update the USER DIRECT file as follows:

```
PROFILE TCPSSLU
  IPL ZCMS PARM FILEPOOL VMSYS
  LOGONBY IBMVM1
  MACH Z
  NAMESAVE TCPIP
```

While the TLS server may support FIPS mode without these changes, IBM makes no claims regarding the cryptographic compliance of an environment outside of the above configuration. These changes will be made part of the IBM-provided user directory in a future release.

MODE NIST-800-131A

The z/VM SSL server can be configured to operate in a mode which meets the requirements of NIST Special Publication 800-131a, which updates guidance on recommended key lengths and usage in a cryptographic environment. This mode of operation changes SSL server operation in the following manner:

- Only TLS 1.2 connections are allowed; all other protocols are automatically disabled.
- Key exchange lengths for RSA, DSA, and Diffie-Hellman must meet a minimum length of 2048.

- Certificates used in a secure connection must meet a minimum hashing requirement of SHA-2. Any certificate hashed with SHA-1 or another hashing algorithm will be rejected.
- Certain cipher suites will automatically be exempted from use. Consult [Table 39 on page 475](#) for information about which cipher suites are considered compliant with NIST SP 800-131A.

MODE NIST-800-131A can be configured only through the MODE operand of the “[VMSSL Command Syntax](#)” on [page 470](#), described later in this chapter.

These cryptographic modes can be used as a means of enforcing more stringent security policies than are allowed by the z/VM SSL server's other options. Note that these modes require advance planning, in terms of certificate database usage, certificate encryption and hashing mechanisms, and allowances for clients that support more modern versions of SSL and TLS. As a result, planning should be done in advance of enabling these options.

Step 1a: Enabling the SSL Server to Use IBM Z Cryptographic Hardware

There are two forms of cryptographic hardware available to IBM Z. The first is the CP-Assisted Cryptographic Facility (CPACF), which is a no-charge feature available on all modern generations of IBM Z and LinuxONE. This feature will accelerate use of symmetric algorithms (such as AES and 3DES) and hashes (such as SHA-1 and SHA-256). It is enabled at the partition level. The TLS/SSL Server will make automatic use of this feature if available, with no configuration required.

The System SSL library supporting the TLS/SSL Server can also offload cryptographic operations to the Crypto Express adapters. These adapters will accelerate asymmetric algorithms (such as clear-key RSA) in addition to the symmetric operations listed previously. In order to enable the TLS/SSL Server to use this function, check the following steps.

1. Verify that Crypto Express queues are available to your z/VM partition. This can be validated by issuing QUERY CRYPTO DOMAINS USERS from an authorized virtual machine. For more information about the QUERY CRYPTO command, see *z/VM: CP Commands and Utilities Reference*. If cryptographic domains are available for shared usage, continue to Step 2. If no domains are available, validate your hardware configuration by checking the LPAR Activation Profile on the Hardware Management Console.
2. Insert the 'CRYPTO APVIRTUAL' statement into the appropriate virtual machine directory entry. This statement will grant the TLS/SSL Server access to shared crypto domains associated with your z/VM partition.
 - If configuring a stand-alone (single) SSLSERV virtual machine, insert this statement directly into the SSLSERV directory entry.
 - If configuring a pool of multiple SSL servers, insert this statement into the appropriate PROFILE directory entry (such as PROFILE TCPSSLU).
3. If necessary, refresh your object directory by using DIRECTXA or appropriate Directory Manager commands.
4. Restart your virtual machine or pool of machines in order to establish access with the virtualized cryptographic hardware. The SSLADMIN RESTART command will reboot some or all of the SSL server machines associated with your TCP/IP stack.
5. When your TLS/SSL servers are running, logged on or disconnected, they will appear in the list of Shared Crypto users available via the QUERY CRYPTO command.

In the unlikely case that cryptographic hardware becomes unavailable to the virtual machine, the TLS/SSL server will automatically fall back to software encryption mechanisms available in the System SSL cryptographic library.

Step 2: Update the TCP/IP Server Configuration File (PROFILE TCPIP)

Include an **SSLSERVERID** statement in the TCP/IP server configuration file, which designates that an SSL server pool is to provide secure communications support. The server pool is initialized automatically when the **SSLSERVERID** statement is processed during TCP/IP initialization.

For example:

```
SSLSERVERID * TIMEOUT 30      ; (Identify use of an SSL server pool; User ID prefix
value                          ; is obtained from DTCPARMS file)
```

Note:

- The prefix for the user IDs that comprise an SSL server pool is determined by a DTCPARMS file **:Nick.** "wildcard" entry for the pool. Do not specify this prefix (or its wildcard variant) as part of the **SSLSERVERID** statement.
- It is recommended that the TCPIP server be allowed to manage initialization of the SSL server pool through use of the **SSLSERVERID** statement only. The use of another mechanism for this purpose (such as the AUTOLOG1 virtual machine) might impact the ability for secured connections to be established as required for your installation.
- Individual SSL server user IDs should not be included in the **AUTOLOG** statement of the TCP/IP server configuration file (PROFILE TCPIP or its equivalent).

In addition, include an **SSLLIMITS** statement, to specify the total number of secure connections allowed and the connection limit for each SSL server that is deployed for your installation.

For example:

```
SSLLIMITS MAXSESSIONS 3000 MAXPERSSLSERVER 600
```

Consult [Chapter 16, “Configuring the TCP/IP Server,”](#) on page 507 for more information about the “**SSLLIMITS** Statement” on page 616 and “**SSLSERVERID** Statement” on page 617.

Step 3: Update the DTCPARMS File for the TCP/IP Server

Include a **:DCSS_Parms.** tag for the TCP/IP server with which the SSL server pool is to provide secure communications support.

For example:

```
:nick.TCPIPTST      :type.server      :class.stack
:DCSS_Parms.<DEFAULT>
...
```

The value **<DEFAULT>** should be specified for this tag, unless your installation requires that specific definitions be used to create the restricted Discontiguous Saved Segment (DCSS) that is used for SSL session cache management.

Also, note that when the **:DCSS_Parms.** entry is encountered by the TCP/IP server initialization program, action is taken to autolog an SSL DCSS Management Agent server (with corresponding shutdown operations performed when the TCP/IP server itself is shutdown). These operations are independent of those performed through processing of any **AUTOLOG** statement in the TCP/IP server configuration file.

For more information about the **:DCSS_Parms.** tag and the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Step 4: Update the DTCPARMS File for the SSL DCSS Management Agent Server

Tags that are significant for the SSL DCSS Management Agent are:

```
:DCSS_Parms.  
:For.  
:Stack.  
:TcpDataFile.
```

For example:

```
:nick.SSTDSSM :type.server :class.ssl_dcsm_agent  
:stack.TCPIPTST  
:TcpDataFile.TCPIPTST DATA  
:for.SST*  
...
```

Note: You should modify DTCPARMS file entries for the SSL DCSS Management Agent if you:

- Require the server to operate on behalf of a TCP/IP stack server different from the default of **TCPIP**. Note that a change to the NAMESAVE statement in the CP directory of the subject SSL DCSS Management Agent likely will be required as well.
- Require the server to operate on behalf of an SSL server pool different from the default of **SSL***.
- Require other server characteristics to be changed, such as the owning user ID.

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Additional notes:

- While the `:DCSS_Parms.` tag (and value) is referenced and used for initialization of the SSL DCSS Management Agent server, this tag must be defined as part of a TCP/IP stack server DTCPARMS entry.
- When an SSL DCSS Management Agent server is deployed for use with a TCP/IP server other than the default of TCPIP, the DTCPARMS `:Type.server.` entry for that SSL server pool must include an appropriate `:Stack.userid` definition. Similarly, an applicable `:TcpDataFile.` tag definition might be required, to ensure the correct TCP/IP data file TCPIPUSERID value is referenced. Otherwise, an association with the correct TCP/IP server will not be established, or, the agent server might not initialize.
- The DTCPARMS `:Type.server.` entry for an SSL DCSS Management Agent also must:
 - include an appropriate `:For.userid` definition. Otherwise, an association with the correct SSL server pool will not be established.
 - be defined in a *nodeID* or SYSTEM DTCPARMS file that resides on the TCP/IP server configuration minidisk (TCPMAINT 198, or its equivalent).
- The DCSS used for session cache information must be unique among each server pool that is defined for use with a given TCP/IP server, it cannot be shared among multiple such server pools. Thus, a separate SSL DCSS Management Agent server must be defined for each SSL server pool that is defined for use on a system.

SSLIDCSS Command

➡ SSLIDCSS ➡

Purpose

The SSLIDCSS command initializes the SSL DCSS Management Agent server, and creates and prepares the SSL server shared session cache for use.

The name and size of the shared session cache DCSS that is created is determined by the DTCPARMS :DCSS_Parms . tag definition that pertains to this agent server (and, its corresponding TCP/IP **stack** server).

Step 5: Update the DTCPARMS File for the SSL Server Pool

When an SSL server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server.

Tags that are significant for an SSL server are:

```
:Admin_ID_list.
:DCSS_Parms.
:Mixedcaseparms.
:Mount.
:Parms.
:OCSPParms.
:Stack.
:TcpDataFile.
:Timestamp.
:Timezone.
:Vmlink.
```

For example:

```
* SSL Server Pool for TCPIPT Test Stack
:nick.SST* :type.server :class.ssl
:stack.TCPIPTST
:tcpdatafile.TCPIPTST DATA
:vmlink. .DIR VMSYS:TCPMAINT.SSLPOOL_SST <. A FORCERW>
:parms.KEYFILE /etc/gskadm/Database.kdb
      PROTO +tlsv1_2
      PROTO -sslv2
```

Note: You should modify DTCPARMS file entries for the SSL server pool if you:

- Require the server pool to operate on behalf of a TCP/IP stack server different from the default of **TCPIP**. Note that a change to the NAMESAVE statement in the CP directory profile of the subject server pool will probably be required as well.
- Require use of a Shared File System directory different from VMSYS:TCPMAINT.SSLPOOL_SSL to be used for server work space.
- Need to set the time zone for the SSL pool servers to match that for your installation.
- Require a Byte File System root file system other than / . . /VMBFS:VMSYS:ROOT/ or are using a file server other than VMSYS.
- Require the SSL server BFS work space to be different from / . . /VMBFS:VMSYS:SSLSERV/.
- Require the SSL certificate database file space to be different from / . . /VMBFS:VMSYS:GSKSSLDB/ or require the mount point for this file space to be other than /etc/gskadm.
- Require the SSL certificate database path name to be different from /etc/gskadm/Database.kdb.
- Change the VMSSL command parameters that are passed to the SSL server. For more information about this command and its operands, see [“VMSSL Command Syntax” on page 470](#).
- Require the use of OCSP or CDP revocation checking of partner certificates used during SSL/TLS connection establishment.
- Require other server characteristics to be changed, such as the owning user ID or to list administrative users that are specific to your installation.

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

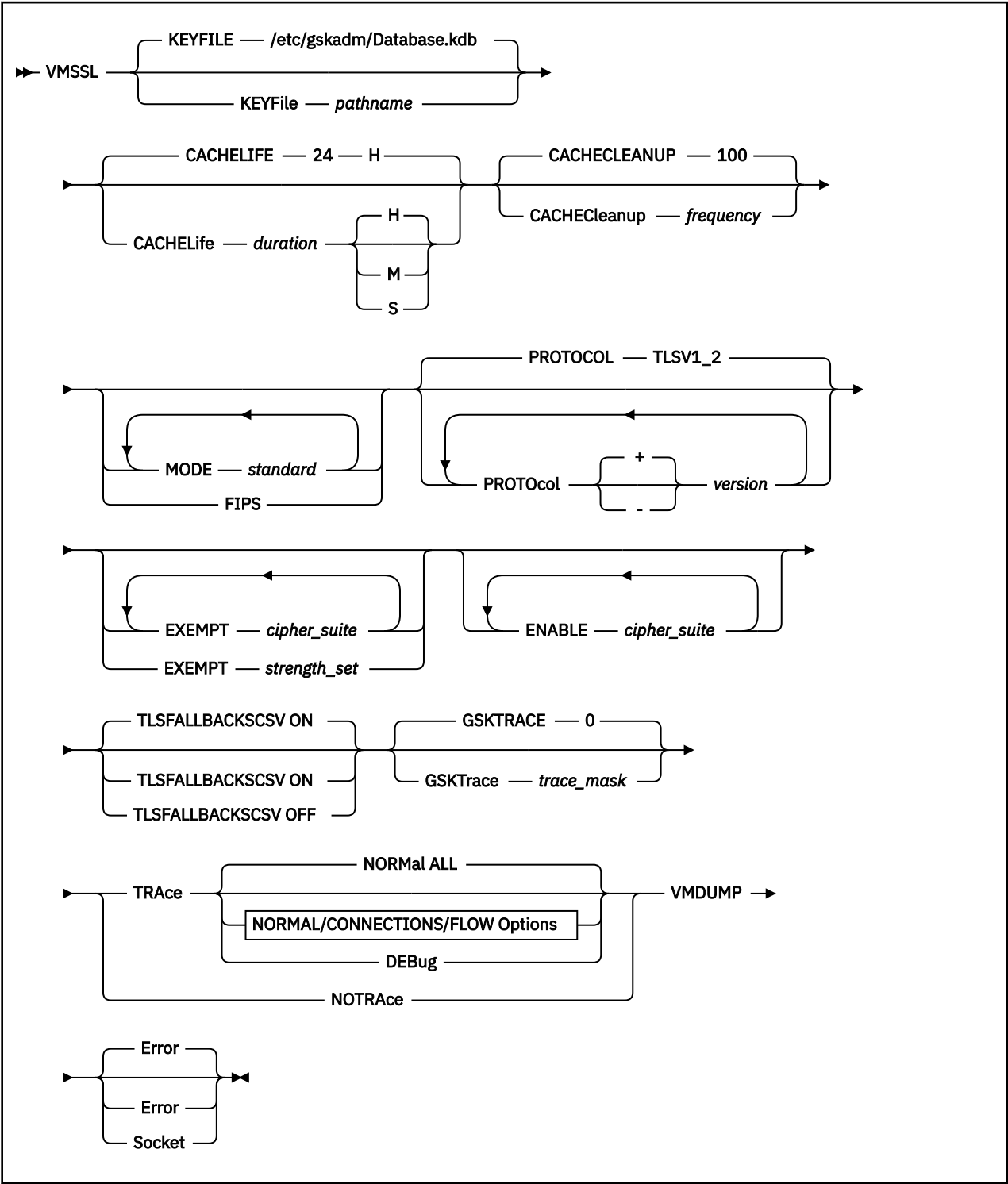
Additional notes:

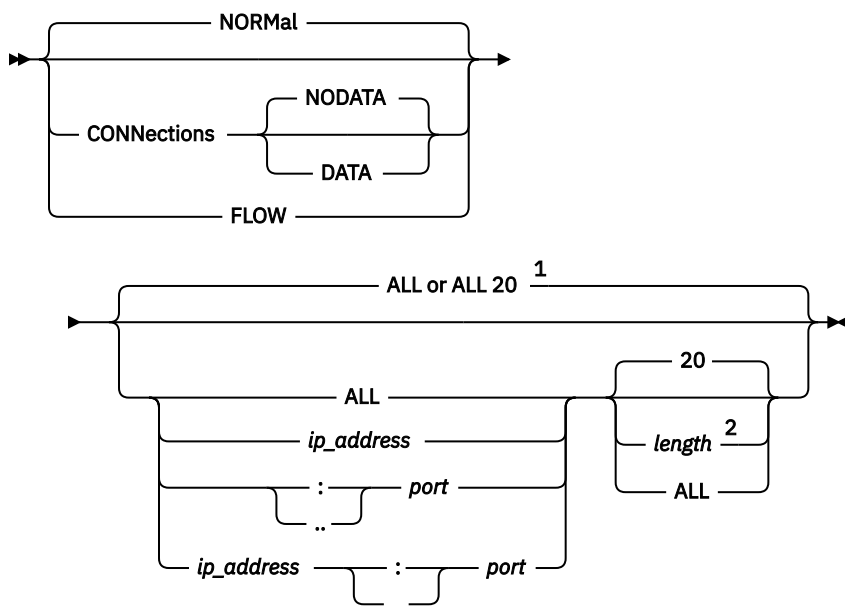
- While the :DCSS_Parms. tag (and value) is referenced and used for initialization of an SSL server, this tag must be defined as part of a TCP/IP stack server DTCPARMS entry.
- When an SSL server is deployed for use with a TCP/IP server other than the default of TCPIP, the DTCPARMS :Type.server. entry for that SSL server pool must include an appropriate :Stack.userid definition. Similarly, an applicable :TcpDataFile. tag definition might be required, to ensure the correct TCP/IP data file TCPIPUSERID value is referenced. Otherwise, an association with the correct TCP/IP server will not be established or the SSL server might not initialize.
- For an SSL server pool, an appropriate :Vmlink. definition must be included as part of the :Type.server. entry for that server pool. This required definition identifies the Shared File System (SFS) directory that is to be used as common-use "work space" by the various member servers. Conversely, a :Vmlink. tag must not be specified for an SSL server that is not defined as a pool server.
- DTCPARMS :Type.server. entries for an SSL server pool must be defined in a *nodeID* or SYSTEM DTCPARMS file that resides on the TCP/IP server configuration minidisk (TCPMAINT 198 or its equivalent).

VMSSL Command Operands (:Parms. Parameters)

VMSSL services are initiated using the VMSSL command. Operands used by this command are obtained from parameters defined by a DTCPARMS file :Parms. tag that is associated with an SSL server pool definition. For more information about this command and its operands, see [“VMSSL Command Syntax”](#) on page 470.

VMSSL Command Syntax



NORMAL/CONNECTIONS/FLOW Options**Notes:**

¹ ALL 20 is the default only when CONNECTIONS DATA is specified. Otherwise, the default is ALL.

² The *length* operand and its default are applicable only when CONNECTIONS DATA is specified.

Operands**KEYFILE *pathname***

specifies the name of the certificate database (key database file or PKCS #12 file) that is to be used by the SSL server. The given name is case sensitive, and can be specified as an absolute or as a relative pathname. The default key database file name is `/etc/gskadm/Database.kdb`

CACHELIFE *duration*

specifies the amount of time (in hours, minutes or seconds) that a session cache entry is to be valid. Cache entries older than the given duration are considered expired and no longer can be used. The minimum duration is 0, and the maximum is 24 hours, which is also the default.

Note: For information about adjusting the CACHELIFE setting, see [“Monitoring the Server Session Cache”](#) on page 486.

H

indicates that *duration* is specified in hours. This is the default.

M

indicates that *duration* is specified in minutes.

S

indicates that *duration* is specified in seconds.

CACHECleanup *frequency*

specifies the *frequency* (as a number of connections) at which the SSL server is to remove expired entries from the shared session cache. The default is to perform clean up operations for every 100 connections that are processed. The minimum *frequency* is 10 and the maximum is 1000.

Note that cleanup operations are performed on a per-server basis. Thus, cleanup might not be performed every *frequency* connections (based on values reported by an SSLADMIN QUERY STATUS or NETSTAT CONFIG SSL command), but every *frequency* connections, as perceived by an individual server.

FIPS

instructs the SSL server to operate in accordance with a particular Federal Information Processing Standard (FIPS) cryptographic standard configuration. This restricts the behavior of the SSL server to approved protocols and cipher suites.

Specifying the FIPS operand is equivalent to specifying the MODE FIPS-140-2 operand.

MODE

establishes a baseline of functionality for the entire SSL server. The values that can be specified for *standard* are:

FIPS-140-2

indicates that the SSL server should operate according to Federal Information Protection Standard (FIPS) 140-2. This mode allows only TLS protocols to be used, and restricts the usage of some cipher suites.

Specifying the MODE FIPS-140-2 operand is equivalent to specifying the FIPS operand.

NIST-800-131a

indicates that the SSL server should operate according to NIST Special Publication 800-131a. This requires the use of TLS 1.2, restricts the usage of certain cipher suites, and mandates the use of RSA, DSA, or Diffie Hellman keys of 2048 or greater for all secure connections.

Note:

- MODE can be specified multiple times to enable available standards of operation. In cases where multiple standards are enabled, the maximum common subset of functionality will be enabled.
- MODE FIPS-140-2 is the preferred method of enabling FIPS-compliant behavior for the SSL server; it should replace use of the FIPS operand.
- If MODE NIST-800-131A is enabled, all protocols other than TLS 1.2 will be automatically disabled.
- Specifying MODE FIPS-140-2 requires that the SSL server be associated with a FIPS-compliant certificate database.

PROTOCOL version

specifies a version of the SSL or TLS protocols to be enabled or disabled for cryptographic use in the operation of this SSL server. The PROTOCOL keyword can be specified multiple times with one version per keyword. Inclusion or exclusion is denoted by a + or - symbol. If no symbol is specified, the default of inclusion is used.

The values that can be specified for *version* are:

TLSV1_2

indicates Transport Layer Security (TLS) 1.2. This is the highest level of cryptographic strength the SSL Server can enable. Certain cipher suites are only available when TLS 1.2 issued. Export ciphers and DES ciphers are excluded from use in TLS 1.2. TLS 1.2 is enabled by default.

TLSV1_1

indicates TLS 1.1. TLS 1.1 is similar to TLS 1.0, though it excludes use of the Export ciphers. TLS 1.1 is disabled by default.

TLSV1_0

indicates TLS 1.0.

SSLV3

indicates Secure Socket Layer (SSL) 3.

SSLV2

indicates SSL 2.

Note:

- PROTOCOL can be specified multiple times. In cases where conflicting instances exist, the most recent instance is used for SSL server processing.
- SSLV2 is incompatible with the TLS protocols. If any level of TLS is specified, SSLV2 is automatically disabled.

- The SSL protocols (SSLV2 and SSLV3) cannot be enabled when running in FIPS mode.
- The SSL server will not initialize if all protocols are disabled.

EXEMPT cipher_suite**EXEMPT strength_set**

specifies a cipher suite, or a set of such suites that have common strength, that should **not** be used by the SSL server.

Note:

- RC4_56_SHA, RC4_EXP1024_56_SHA, and DES_EXP1024_56_SHA were accepted as valid cipher suite names in releases prior to z/VM 5.4. They are no longer supported.
- For cipher suite names, V2 and V3 designate the version of SSL or TLS used by the SSL Server. In cases where a cipher suite is V2 and V3 compliant, specifying that suite name for exemption exempts V2 and V3.
- Inclusion or exclusion of specific versions of SSL or TLS through the PROTOCOL operand can change the cipher suites available to the SSL server, even when using EXEMPT on a common-strength basis.
- The following cipher suites are disabled by default:

V3 Code Name

```

0000 - NULL
0001 - NULL_MD5
0002 - NULL_SHA
0003 - RC4_40_MD5
0004 - RC4_128_MD5
0005 - RC4_128_SHA
0006 - RC2_40_MD5
0009 - DES_56_SHA
000A - 3DES_168_SHA
000C - DH_DSS_DES
000D - DH_DSS_3DES
000F - DH_RSA_DES
0010 - DH_RSA_3DES
0012 - DHE_DSS_DES
0015 - DHE_RSA_DES
0030 - DH_DSS_AES_128
0031 - DH_RSA_AES_128
0036 - DH_DSS_AES_256
0037 - DH_RSA_AES_256
0068 - DH_DSS_AES_256_SHA256
0069 - DH_RSA_AES_256_SHA256
003B - NULL_SHA256
003E - DH_DSS_AES_128_SHA256
003F - DH_RSA_AES_128_SHA256
00A0 - DH_RSA_AES_128_GCM_SHA256
00A1 - DH_RSA_AES_256_GCM_SHA384
00A4 - DH_DSS_AES_128_GCM_SHA256
00A5 - DH_DSS_AES_256_GCM_SHA384
C001 - ECDH_ECDSA_NULL_SHA
C002 - ECDH_ECDSA_RC4_128_SHA
C003 - ECDH_ECDSA_3DES_EDE_SHA
C004 - ECDH_ECDSA_AES_128_SHA
C005 - ECDH_ECDSA_AES_256_SHA
C007 - ECDHE_ECDSA_RC4_128_SHA
C00B - ECDH_RSA_NULL_SHA
C00C - ECDH_RSA_RC4_128_SHA
C00D - ECDH_RSA_3DES_EDE_SHA

```

C00E - ECDH_RSA_AES_128_SHA
 C00F - ECDH_RSA_AES_256_SHA
 C010 - ECDHE_RSA_NULL_SHA
 C011 - ECDHE_RSA_RC4_128_SHA
 C025 - ECDH_ECDSA_AES_128_SHA256
 C026 - ECDH_ECDSA_AES_256_SHA384
 C029 - ECDH_RSA_AES_128_SHA256
 C02A - ECDH_RSA_AES_256_SHA384
 C02D - ECDH_ECDSA_AES_128_GCM_SHA256
 C02E - ECDH_ECDSA_AES_256_GCM_SHA384
 C031 - ECDH_RSA_AES_128_GCM_SHA256
 C032 - ECDH_RSA_AES_256_GCM_SHA384

- Management by strength:
 - Disallows modifications on a per-cipher basis
 - Works only for the cipher suites that are enabled by default for the TLS/SSL server
 - Should not be used in environments with complex or rapidly changing encryption requirements

ENABLE *cipher_suite*

specifies a cipher suite that should be used by the SSL server.

Notes:

- ENABLE can be used only when managing cipher suites by name. ENABLE cannot be used in conjunction with the strength keywords (LOW or NONE, for example).
- Like EXEMPT, ENABLE cannot be used if EXEMPT processing is being handled by strength.
- If the same cipher suite is specified for ENABLE and EXEMPT:
 1. EXEMPT processing is handled first, regardless of DTCPARMS ordering in releases prior to z/VM 6.3.
 2. The most recent instance is used for SSL server processing in z/VM 6.3 and later.

The possible values for *cipher_suite* follow in [Table 38 on page 474](#) and [Table 39 on page 475](#):

<i>Table 38. SSLV2 Cipher Suite Values</i>			
Name	Strength	Key Length	V2 Code
RC2_40_MD5	Low	40	4
RC4_40_MD5	Low	40	2
DES_56_SHA	Low	56	6
RC2_128_MD5	Medium	128	3
RC4_128_MD5	Medium	128	1
3DES_168_SHA	High	168	7

Table 39. SSLV3 and TLS Cipher Suite Values

Name	Strength	Symmet- ric Key Length	Cipher Code	Mode FIPS- 140-2	Mode NIST- 800- 131A	Enabled by Default
NULL	N/A	None	0000			
NULL_SHA	N/A	None	0002			
NULL_SHA256	N/A	None	003B			
NULL_MD5	N/A	None	0001			
RC2_40_MD5	N/A	40	0006			
RC4_40_MD5	N/A	40	0003			
DES_56_SHA	Low	56	0009			
DH_DSS_DES	N/A	56	000C			
DH_RSA_DES	N/A	56	000F			
DHE_DSS_DES	Low	56	0012			
DHE_RSA_DES	Low	56	0015			
RC4_128_SHA	N/A	128	0005			
RC4_128_MD5	N/A	128	0004			
RSA_AES_128	Medium	128	002F	Y	Y	Y
RSA_AES_128_SHA256	Medium	128	003C	Y	Y	Y
RSA_AES_128_GCM_SHA256	High	128	009C	Y	Y	Y
DH_DSS_AES_128	N/A	128	0030	Y		
DH_DSS_AES_128_SHA256	N/A	128	003E	Y		
DH_RSA_AES_128	N/A	128	0031	Y	Y	
DH_RSA_AES_128_SHA256	N/A	128	003F	Y	Y	
DH_DSS_AES_128_GCM_SHA256	N/A	128	00A4	Y	Y	
DH_RSA_AES_128_GCM_SHA256	N/A	128	00A0	Y	Y	
DHE_DSS_AES_128	Medium	128	0032	Y		Y
DHE_DSS_AES_128_SHA256	Medium	128	0040	Y		Y
DHE_DSS_AES_128_GCM_SHA256	High	128	00A2	Y	Y	Y
DHE_RSA_AES_128	Medium	128	0033	Y	Y	Y
DHE_RSA_AES_128_SHA256	Medium	128	0067	Y	Y	Y
DHE_RSA_AES_128_GCM_SHA256	High	128	009E	Y	Y	Y
3DES_168_SHA	High	168	000A	Y		
DH_DSS_3DES	N/A	168	000D	Y		
DH_RSA_3DES	N/A	168	0010	Y	Y	
DHE_DSS_3DES	N/A	168	0013	Y		

Table 39. SSLV3 and TLS Cipher Suite Values (continued)

Name	Strength	Symmet- ric Key Length	Cipher Code	Mode FIPS- 140-2	Mode NIST- 800- 131A	Enabled by Default
DHE_RSA_3DES	N/A	168	0016	Y	Y	
RSA_AES_256	High	256	0035	Y	Y	Y
RSA_AES_256_SHA256	High	256	003D	Y	Y	Y
RSA_AES_256_GCM_SHA384	High	256	009D	Y	Y	Y
DH_DSS_AES_256	N/A	256	0036	Y		
DH_DSS_AES_256_SHA256	N/A	256	0068	Y		
DH_DSS_AES_256_GCM_SHA384	N/A	256	00A5	Y	Y	
DH_RSA_AES_256	N/A	256	0037	Y	Y	
DH_RSA_AES_256_SHA256	N/A	256	0069	Y	Y	
DH_RSA_AES_256_GCM_SHA384	N/A	256	00A1	Y	Y	
DHE_DSS_AES_256	High	256	0038	Y		Y
DHE_DSS_AES_256_SHA256	High	256	006A	Y		Y
DHE_DSS_AES_256_GCM_SHA384	High	256	00A3	Y	Y	Y
DHE_RSA_AES_256	High	256	0039	Y	Y	Y
DHE_RSA_AES_256_SHA256	High	256	006B	Y	Y	Y
DHE_RSA_AES_256_GCM_SHA384	High	256	009F	Y	Y	Y
ECDH_ECDSA_NULL_SHA	None	None	C001			
ECDH_ECDSA_RC4_128_SHA	Medium	128	C002			
ECDH_ECDSA_3DES_EDE_SHA	Medium	168	C003	Y		
ECDH_ECDSA_AES_128_SHA	High	128	C004	Y		
ECDH_ECDSA_AES_256_SHA	High	256	C005	Y		
ECDHE_ECDSA_NULL_SHA	None	None	C006			
ECDHE_ECDSA_RC4_128_SHA	Medium	128	C007			
ECDHE_ECDSA_3DES_EDE_SHA	Medium	168	C008	Y		Y
EDCHE_ECDSA_AES_128_SHA	High	128	C009	Y		Y
ECDHE_ECDSA_AES_256_SHA	High	256	C00A	Y		Y
ECDH_RSA_NULL_SHA	None	None	C00B			
ECDH_RSA_RC4_128_SHA	Medium	128	C00C			
ECDH_RSA_3DES_EDE_SHA	Medium	168	C00D	Y		
ECDH_RSA_AES_128_SHA	High	128	C00E	Y		
ECDH_RSA_AES_256_SHA	High	256	C00F	Y		
ECDHE_RSA_NULL_SHA	None	None	C010			

Table 39. SSLV3 and TLS Cipher Suite Values (continued)

Name	Strength	Symmet- ric Key Length	Cipher Code	Mode FIPS- 140-2	Mode NIST- 800- 131A	Enabled by Default
ECDHE_RSA_RC4_128_SHA	Medium	128	C011			
ECDHE_RSA_3DES_EDE_SHA	Medium	168	C012	Y		Y
ECDHE_RSA_AES_128_SHA	High	128	C013	Y		Y
ECDHE_RSA_AES_256_SHA	High	256	C014	Y		Y
ECDHE_ECDSA_AES_128_SHA256	High	128	C023	Y	Y	Y
ECDHE_ECDSA_AES_256_SHA384	High	256	C024	Y	Y	Y
ECDH_ECDSA_AES_128_SHA256	High	128	C025	Y	Y	
ECDH_ECDSA_AES_256_SHA384	High	256	C026	Y	Y	
ECDHE_RSA_AES_128_SHA256	High	128	C027	Y	Y	Y
ECDHE_RSA_AES_256_SHA384	High	256	C028	Y	Y	Y
ECDH_RSA_AES_128_SHA256	High	128	C029	Y	Y	
ECDH_RSA_AES_256_SHA384	High	256	C02A	Y	Y	
ECDHE_ECDSA_AES_128_GCM_SHA256	High	128	C02B	Y	Y	Y
ECDHE_ECDSA_AES_256_GCM_SHA384	High	256	C02C	Y	Y	Y
ECDH_ECDSA_AES_128_GCM_SHA256	High	128	C02D	Y	Y	
ECDH_ECDSA_AES_256_GCM_SHA384	High	256	C02E	Y	Y	
ECDHE_RSA_AES_128_GCM_SHA256	High	128	C02F	Y	Y	Y
ECDHE_RSA_AES_256_GCM_SHA384	High	256	C030	Y	Y	Y
ECDH_RSA_AES_128_GCM_SHA256	High	128	C031	Y	Y	
ECDH_RSA_AES_256_GCM_SHA384	High	256	C032	Y	Y	

The possible values for *strength_set* are:

LOW

specifies that all of the ciphers listed in Table 38 on page 474 and Table 39 on page 475 with a strength designation of **LOW** are to be exempt from use by the SSL server.

MEDIUM

specifies that all of the ciphers listed in Table 38 on page 474 and Table 39 on page 475 with a strength designation of **MEDIUM** are to be exempt from use by the SSL server.

Strength set names can be specified in abbreviated form, with the first character of each accepted as the minimum value.

Note:

- Cipher suite NULL provides no security. Exempting all cipher suites **except** NULL means that no data is encrypted.
- The SSL server will not initialize if all cipher suites are exempted.
- When cipher suites are exempted based on a common-strength basis, all cipher suites of lower strength than that specified are also exempted.

- The set of high-strength cipher suites cannot be exempted on a common-strength basis, since doing so would exempt *all* cipher suites from use by the SSL server.
- The EXEMPT operand can be used to exempt a set of cipher suites based on common strength or repeated to exempt specific, named cipher suites. Set names and individual cipher suite names cannot be combined with one another.

TLSFallbackSCSV

instructs the TLS/SSL server to act in accordance with RFC 7507. This option enables protection against protocol downgrade attacks. ON is enabled by default.

GSKTRACE *trace_mask*

specifies that detailed SSL trace information should be captured by the SSL server. The type of the information captured is controlled by a bit mask that is determined by the supplied trace mask value, which can be specified as a decimal (nnn), octal (0nnn), or hexadecimal (0xhh) value. No trace option is enabled if the bit mask is 0 or if the GSKTRACE operand is omitted, and all trace options are enabled if the trace mask is 0xffff.

The following trace mask options are available:

0x01 = Trace function entry
0x02 = Trace function exit
0x04 = Trace errors
0x08 = Include informational messages
0x10 = Include EBCDIC data dumps
0x20 = Include ASCII data dumps

These options can be combined so that the desired tracing is performed. For example, to trace only errors and include informational messages, specify the GSKTRACE operand as one of the following:

```
GSKTRACE 12  
GSKTRACE 014  
GSKTRACE 0x0C
```

SSL trace information is written to the BFS file `/tmp/user_id.gskssl.%.trc` where *user_id* is replaced by the SSL server user ID, and the percent sign (%) is replaced with a numeric process identifier. For example, if the user ID of the SSL server is SSL00001 and the process identifier in effect is 247, the trace file produced will be: `/tmp/ss100001.gskssl.247.trc`.

The GSKTRACE command creates a readable copy of SSL trace information. For more information, consult "[SSL Tracing Information](#)" in *z/VM: TCP/IP User's Guide*.

NOTRAce

specifies that all tracing is turned off. This is the default.

TRAcce

specifies that tracing is to be performed. The TRACE operand is intended for use in diagnosing SSL server operational problems, in consultation with the IBM support center. Use of this operand to perform detailed SSL server tracing is strongly discouraged in production environments.

NORMal

specifies that a trace entry is recorded to indicate a successful connection. This is the default if TRACE is specified.

CONNECTIONS

specifies that a trace entry is recorded for connection state changes and handshake results.

NODATA

specifies that no data is displayed for send and receive trace entries. This is the default if CONNECTIONS is specified.

DATA

specifies that the first 20 bytes of data are displayed for send and receive trace entries.

FLOW

specifies that flow of control and system activity are traced.

DEBug

specifies that extensive tracing is done for all control and system activity as well as data on all connections.

ALL

specifies that tracing is done for all connections. This is the default if TRACE is specified.

ip_address

specifies that tracing is done only for activity associated with this IP address.

:port**..port**

specifies that tracing is done only for activity associated with this port.

Note: The format *:port* is not valid with IPv6 addresses, use *..port* instead.

length

specifies the number of bytes of data to be presented when the CONNECTIONS DATA operand is used. The connection data is represented in hexadecimal, as well as in ASCII and EBCDIC, in unencrypted form. The length must be specified as 0, or as a number in the range of 1 to 65535. The value zero (0) or the keyword ALL indicates that all available data is to be presented. The default is to display 20 bytes of data. Note that a suitable tracing target (such as an IP address, port, or connection number) must be designated when a length value other than the default is to be used.

VMDUMP error_type

instructs the SSL server to create a virtual machine dump when an error of the indicated type is encountered. In addition, the affected server initiates the creation of dumps for its associated TCP/IP stack and DCSS agent servers, when conditions allow for this.

error_type

identifies the type of errors for which a virtual machine dump is to be created. Possible values for 'error_type' are:

Error

specifies that a dump is to be created for an unexpected severe error condition. This is the default.

Socket

specifies that a dump is to be created for unexpected socket-related errors only.

Note:

- The SSL server (or server pool) requires authorization to use the non-general version of the CP FOR command when the VMDUMP operand is specified. IBM-defined privilege class C provides this authorization.
- The virtual machine dumps created by using the VMDUMP operand are processed using the SYSTEM operand of the CP VMDUMP command (thus, dumps are transferred to the user specified on the SYSTEM_USERIDS CP configuration statement of the SYSTEM CONFIG file).

Usage Notes

1. DTCPARMS file changes become effective only when the SSL server is restarted.
2. Certain informational messages are always displayed at the SSL server console to:
 - acknowledge the receipt of SSLADMIN commands
 - report potential security breaches, such as a message digest not matching the message during the handshake
3. A key database that is created as a FIPS mode database, can only be updated by **gskkyman** or by using the CMS APIs executing in FIPS mode. Such a database, however, may be opened as read-only when executing in non-FIPS mode. Key databases created while in non-FIPS mode cannot be opened when executing in FIPS mode. For additional FIPS mode information and considerations, consult [SSL Certificate/Key Management and SSL Tracing Information](#) in *z/VM: TCP/IP User's Guide*.

4. To use a PKCS #12 file in FIPS mode, the file must be protected using TDES. When creating a PKCS #12 file from certificates within a key database file, using the **gskkyman** utility, the key database must be a FIPS key database.
5. For information about trace output, see the [*z/VM: TCP/IP Diagnosis Guide*](#).

Messages

- DTCSSL2461W The SSLv2 protocol is disabled when TLS is enabled
- DTCSSL2462W Only TLS protocols can be enabled in FIPS mode; SSL protocols have been disabled

Step 6: Set Up the Certificate (Key) Database

Before the SSL server can be used to secure any connections, an SSL certificate database (or, key database) must exist, and must be populated with one or more server certificates.

Only two kinds of certificate database are allowed as SSL certificate database, one is the standard key database file which has a file extension of .kdb, another is the PKCS #12 certificate store which has a file extension of .p12 or .pfx. SSL server will not support the other certificate database file name.

Use the steps that follow to create and prepare the key database file:

1. Log on the **GSKADMIN** user ID and allow its default PROFILE EXEC to run.

For convenience, an IBM-supplied PROFILE EXEC has been provided for the GSKADMIN user ID that prepares a basic work environment for performing certificate database management tasks through use of the **gskkyman** utility program. If necessary this PROFILE EXEC can be customized to meet local system requirements or account for use of Byte File System (BFS) directories other than the IBM system deliverable defaults.

2. Invoke the **gskkyman** utility. After the top-level menu is displayed, select the **Create new database** function and provide appropriate data and responses to the prompts that are presented. As part of this task, specify the database name default (**Database.kdb**) and a database password that conforms to the guidelines established for your installation.

Note:

- Passwords associated with the key database and certificates are case sensitive. Be certain the password is entered as desired and is one that can adequately be recalled.
- When a null response is provided during interaction with the **gskkyman** utility, **enter** must be used twice.

```

Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

0 - Exit program

Enter option number:
1 <enter>

Enter key database name (press ENTER to return to menu):
Database.kdb <enter>
Enter database password (press ENTER to return to menu):
<enter password>
Re-enter database password:
<enter password>

Enter password expiration in days (press ENTER for no expiration):
35 <enter>

Enter database record length (press ENTER to use 5000):
<enter>
<enter>

Enter 1 for FIPS mode database or 0 to continue:
0 <enter>

Key database /etc/gskadm/Database.kdb created.

Press ENTER to continue.
<enter>
<enter>

```

3. Store the database password (in a predefined password "stash" file) to allow for use of key database by the SSL server.

```

Key Management Menu

Database: /etc/gskadm/Database.kdb
Expiration: None

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
10 <enter>

Database password stored in /etc/gskadm/Database.sth.

Press ENTER to continue.
<enter>
<enter>

```

4. Exit the **gskkyman** program.

```

Key Management Menu

Database: /etc/gskadm/Database.kdb
Expiration: None

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
0 <enter>

```

5. Issue the **OPENVM** commands that follows to confirm that the necessary database files have been created, and to list the permissions of these files.

openvm list /etc/gskadm/

```

Directory = '/etc/gskadm/'
Update-Dt Update-Tm Type Links Bytes Path name component
09/12/2008 18:50:47 F 1 65080 'Database.kdb'
09/12/2008 18:51:21 F 1 80 'Database.rdb'
09/12/2008 18:51:01 F 1 129 'Database.sth'
Ready; T=0.01/0.01 18:51:34

```

openvm list /etc/gskadm/ (own)

```

Directory = '/etc/gskadm/'
User ID Group Name Permissions Type Path name component
gskadmin security rw- --- --- F 'Database.kdb'
gskadmin security rw- --- --- F 'Database.rdb'
gskadmin security rw- --- --- F 'Database.sth'
Ready; T=0.01/0.01 18:51:37

```

6. Issue the **OPENVM PERMIT** commands that follow to allow the SSL server to access the newly-created key database:

openvm permit /etc/gskadm/Database.kdb rw- r-- ---

openvm permit /etc/gskadm/Database.sth rw- r-- ---

7. Issue the **OPENVM LIST** command that follows to confirm that **r** (read) has been added to the "group" permissions for the key database and password stash files:

openvm list /etc/gskadm/ (own)

```

Directory = '/etc/gskadm/'
User ID Group Name Permissions Type Path name component
gskadmin security rw- r-- --- F 'Database.kdb'
gskadmin security rw- --- --- F 'Database.rdb'
gskadmin security rw- r-- --- F 'Database.sth'
Ready; T=0.01/0.01 18:55:15

```

With the key database now in place, the SSL server can be initialized to confirm it has access to this database.

The PKCS #12 file could be created through the Key Management menu of the **gskkyman** utility and is placed in the BFS directory, it will be introduced below. Besides, a password file which has a file extension of .p12pw needs to be created to store the password of the PKCS #12 file and is placed in the same BFS directory. OPENVM PERMIT command should be used to grant read access to them to allow the SSL server to access them.

Note: Do not attempt to logon the SSL server via a secure Telnet connection. For more information, see “TCP/IP and SSL Server Logon Restrictions” on page 52.

Note that the z/VM system deliverable defines the GSKADMIN user ID as a class G user, so it does not have authorization to XAUTOLOG the SSL server. One can use the TCPMAINT user ID for this purpose, or manually logon the SSL server to verify the key database is accessible.

The key database now can be populated with the appropriate server and CA certificates required to provide SSL-protected communications for your installation. For more information about this task, see *z/VM: TCP/IP User's Guide*, *SSL Certificate Management*, under the subheading "Key Management Menu".

To create a self-signed certificate for basic testing, follow the instructions provided under the subheading "Create a Self-Signed Server or Client Certificate". The resulting self-signed certificate can be treated as though it was signed by an internal CA and exported to other SSL servers and client applications.

Note: To use a self-signed certificate in a key database other than the z/VM key database in which it was created, the certificate must be exported with its private key, in PKCS #12 format. This can be accomplished through the "Key Management" menu of the **gskkyman** utility (as cited above), using the following sequence of selections:

```
1 - Manage keys and certificates
7 - Export certificate and key to a file
3 - Binary PKCS #12 Version 3
```

Example responses to applicable export command prompts follow:

```
...
Enter export file name (press ENTER to return to menu):
yourcert.arm <enter>

Enter export file password (press ENTER to return to menu):
<enter password>
Re-enter export file password:
<enter password>

Enter 1 for strong encryption, 0 for export encryption:
1 <enter>
...
```

The exported certificate is now available in the /etc/gskadm BFS directory. It then can be propagated to an accessed z/VM minidisk (or SFS directory) by issuing an appropriate **OPENVM GETBFS** command. For example:

```
openvm getbfs /etc/gskadm/yourcert.arm yourcert arm a
```

When certificates are exported from the key database to a BFS file using a **binary** file format, via either of these **gskkyman** export options:

```
1 - Binary PKCS #12 Version 1
3 - Binary PKCS #12 Version 3
```

the resulting file, when propagated to a minidisk, should be processed with the **OPENVM GETBFS** command with the **(BFSLINE NONE)** option to maintain the binary nature of the file.

Conversely, when certificates are exported from the key database to a BFS file using a **Base64** file format, via either of these **gskkyman** export options:

```
2 - Base64 PKCS #12 Version 1
4 - Base64 PKCS #12 Version 3
```

the resulting file, when propagated to a minidisk, should be processed with the **OPENVM GETBFS** command with the **(BFSLINE NL)** option to ensure the appropriate record structure is maintained.

Note that attempts to import an incorrect exported certificate into another certificate database likely will fail, and might be reported as one of the following types of error conditions:

- The certificate password is not valid

- The certificate content is not valid
- The certificate length is not valid

To import a certificate with its private key into a new database, whether it is a certificate previously exported using **gskkyman** or an acquired certificate that has been placed into the `/etc/gskadm` BFS directory via an `OPENVM PUTBFS` command, one should access the **gskkyman** "Key Management" menu for the key database in use, and select the "Import a certificate and a private key" menu. From this menu, you are prompted to enter the name of the certificate file. The subject file should be in the `/etc/gskadmin` BFS directory and must be in PKCS #12 file format (usually, with a filename that ends with the string **.p12**).

z/VM Certificate Label Requirements	
The labels for certificates that are to be used by the SSL server (whether those certificates are server certificates or self-signed certificates) must be no more than eight characters, and must be comprised of only uppercase, alphanumeric characters.	

Specify a unique label that conforms to the requirements for a z/VM certificate label when the certificate is imported into the key database. The label then can be specified as SSL configuration changes are implemented for your z/VM system.

Notes: In the *z/VM: TCP/IP User's Guide*, see:

- [Acting As Your Own Certificate Authority \(CA\)](#), if you intend to act as your own certificate authority.
- [CERTMGR Command](#), if you would like to query certificates within a specific certificate database.

Step 7: Implement Customization for Protected Communications

If your installation is to support static (implicit) SSL connections, you must update the TCP/IP server configuration file to designate your secure ports and to associate those ports with the certificates you have stored in the certificate database.

If your installation is to support Dynamic (explicit) TLS/ SSL connections, you must update the appropriate client and server configuration files so that TLS connections can be accommodated. Consult the respective client and server configuration chapters in this and other TCP/IP publications for more information about the changes required to support TLS connections.

Note:

1. The static and dynamic mechanisms cannot both be used to provide secure communications for a given port or service. If such a configuration is attempted, the static mechanism has precedence and is used for all connections. Results for dynamically-secured connection attempts for a service configured in this manner are unpredictable.
2. The LDAP server makes use of SSL/TLS services that are separate from those provided by the z/VM SSL server. Thus, you do not protect the LDAP server ports in the same manner as that described here, for other servers such as the FTP server.
3. For testing purposes, you can specify the label of a self-signed certificate instead of a server certificate to be used for protecting communications associated with a designated port or service. For more information, see *z/VM: TCP/IP User's Guide*.

When Telnet ports (those associated with the INTCLIEN server) other than the default port of 23 are to be protected, the PORT operand of the INTERNALCLIENTPARMS statement must be updated to include such port numbers. If this change is not made, the Telnet server will not accept connection requests that are made to those ports.

Step 7A. Designate the Secure Ports (Static SSL Connections)

To designate a secure port for a TCP/IP server, you must include the SECURE operand on the PORT statement that reserves that port for the server. When a port is designated as SECURE, a connection request from a client to the server that listens on that port is routed by TCP/IP to the SSL server. If

SECURE is not specified, the port is considered not secure, and connection requests do not involve SSL processing.

In addition to the SECURE operand, the PORT statement for a secure port must include the label of the server certificate to be used for secure connections to that port.

After you designate the secure ports in the TCP/IP server configuration file, you must activate these changes in one of the following ways:

- To make the changes permanent, you must restart the TCP/IP (stack) server.
- To activate the changes dynamically, use the OBEYFILE command. For more information, see [“OBEYFILE Command” on page 644](#).

Note: It is not recommended to configure a port for the FTP server to use static SSL, as in such case the server will not comply to the FTP protocol standard. See [“Step 6: Configure Secure FTP Connections \(Optional\)” on page 69](#) for information on how to secure FTP communications with SSL.

More Information

- [“PORT Statement” on page 603](#)
- [“INTERNALCLIENTPARMS Statement” on page 584](#)

Step 7B. Configure TLS Services (Dynamic SSL/TLS Connections)

Configuring a specific service to use SSL/TLS-protected communications requires the use of one or more configuration file statements that are associated with the server that provides the service of interest. Consult the respective client and server configuration chapters in this and other TCP/IP publications for more information about the changes required to support TLS connections.

More Information

- [“Step 6: Configure Secure FTP Connections \(Optional\)” on page 69](#)
- [“Configuring the Server for Secure SMTP” on page 406](#)
- [“INTERNALCLIENTPARMS Statement” on page 584](#)
- [“SECRETELNETCLIENT statement” on page 21](#)
- [“FTP DATA File Statements” in *z/VM: TCP/IP User's Guide*](#)
- [“Transferring Files Using Secure FTP in TCP” in *z/VM: TCP/IP User's Guide*](#)

Dynamic Server Operation

The SSL server provides an SSLADMIN command interface that allows you to perform server administration tasks. Detailed descriptions of the SSLADMIN commands available for use are provided later.

Note: To use SSLADMIN commands, the server code disk (TCPMAINT 591) must be accessed, and the invoking user ID must be designated as an SSL server administrative user, via an appropriate DTCPARMS file :Admin_ID_List. tag definition.

SSL Server Administration

This section describes how to perform various SSL server administration tasks, such as starting or stopping the server and turning tracing on or off.

Starting an SSL Server

To start an SSL server or server pool, issue either an SSLADMIN START command or a NETSTAT SSL START command. For example:

```
ssladmin start (ssl ssl00004
```

or perhaps:

```
netstat ssl start all
```

Note: Do not attempt to logon an SSL server via a secure Telnet connection. For more information, see [“TCP/IP and SSL Server Logon Restrictions” on page 52.](#)

Stopping the Server

To stop an SSL server or server pool, issue the SSLADMIN STOP command:

```
ssladmin stop (ssl ssl00004
```

or perhaps:

```
ssladmin stop (ssl all
```

Accommodating Changes to the Key Database

When the **gskkyman** utility is used to modify the content of the SSL key database while the SSL server is active, use the **SSLADMIN REFRESH** command to instruct the server to obtain updated key database information:

```
ssladmin refresh
```

Monitoring the Server Session Cache

It is possible for the security parameters — negotiated with a client during a previous secure session — to be reused. Resuming a previously negotiated session with such parameters can reduce network traffic and CPU time, and therefore can be desirable. However, note that the resumption of a session can be initiated only by a client, and not by the SSL server.

When a client chooses to resume a session, it supplies the session ID of the session that is to be resumed. If the SSL server finds this ID in its session cache, it returns the same session ID to the client. Each side then sends an encrypted message as a test. If this test is successful, no further negotiations are needed, and data flow begins. If the SSL server cannot find the requested session ID in its cache, the entry might have expired (exceeded the CACHELIFE setting), or the entry might not have been added to the cache, due to capacity restraints or an unexpected caching problem.

SSL server cache usage can be checked by issuing the SSLADMIN QUERY CACHE command. When responses for this command are reviewed, keep in mind that the resumption of a secure session using cached information is not a common occurrence. Thus, a response that cites only a small number of resumed sessions (and perhaps, no such sessions) would not be unusual. However, if conditions exist where extensive use of session cache information is anticipated or expected, the values for the CACHECLEANUP and CACHELIFE operands of the VMSSL command can be adjusted to affect the management of session cache information. For more information, see [“VMSSL Command Syntax” on page 470.](#)

In addition, the number of cache overruns reported in an SSLADMIN QUERY CACHE response will likely be reported to be low, or often zero — again, due to infrequent use of resumed sessions using cache information, and also due to the (default) size of the Discontiguous Saved Segment (DCSS) that is used to maintain session cache information. In the event that a substantial number of cache overruns are reported, the size of the Discontiguous Saved Segment (DCSS) that is used to maintain session cache information, might need to be modified. For more information, see [“Step 4: Update the DTCPARMS File for the SSL DCSS Management Agent Server” on page 467.](#)

Tracing Server Activities

To begin tracing SSL server activities when the server starts, use the TRACE startup parameter on the :Parms. tag:

- You can specify TRACE as a :Parms. tag startup parameter in the DTCPARMS file. Tracing will begin each time the server is autologged until you modify the DTCPARMS file again to remove the TRACE startup parameter. See [“Step 3: Update the DTCPARMS File for the TCP/IP Server”](#) on page 466.
- You can specify TRACE on the :Parms. tag when manually starting the server while logged on to the SSL server virtual machine. See [“VMSSL Command Syntax”](#) on page 470.

To start or stop tracing SSL server activities dynamically while the SSL server is running, use the SSLADMIN TRACE/NOTRACE command. See [“SSLADMIN TRACE/NOTRACE Command”](#) on page 500.

For information about trace output, see the [z/VM: TCP/IP Diagnosis Guide](#).

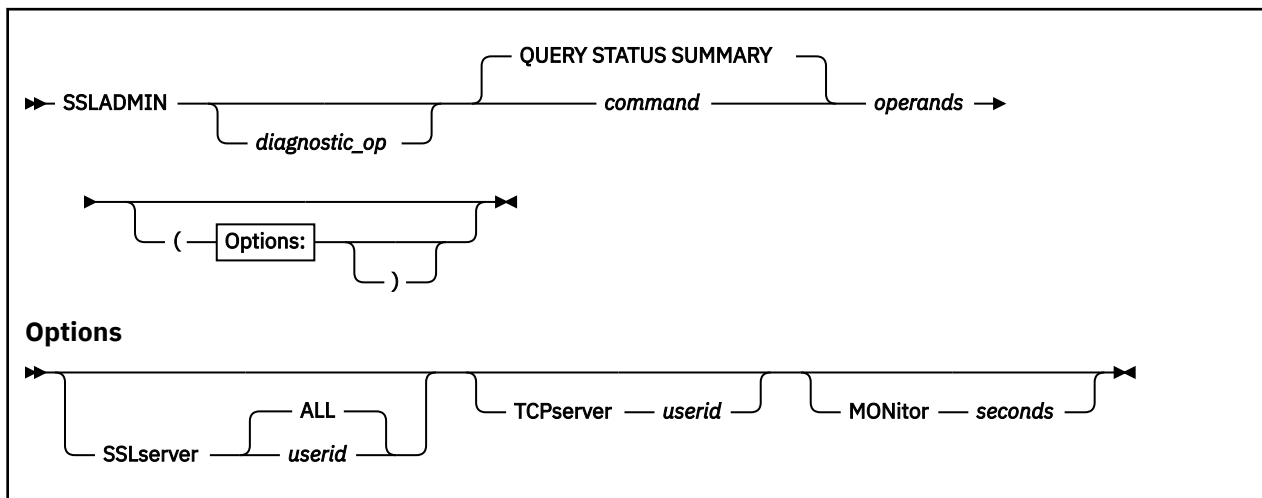
SSL Server Administration Commands

Table 40 on page 487 provides a summary of SSL server administration commands.

Table 40. SSL Administration Commands

Command	Description	Location
SSLADMIN	General SSLADMIN command format.	“General SSLADMIN Command” on page 488
SSLADMIN CLEAR	Removes a previously established server user ID default by an SSLADMIN SET command.	“SSLADMIN CLEAR Command” on page 490
SSLADMIN CLOSECON	Closes the SSL server console log and sends console log to the server owner.	“SSLADMIN CLOSECON Command” on page 490
SSLADMIN HELP	Displays a list and brief explanation of SSLADMIN commands.	“SSLADMIN HELP Command” on page 490
SSLADMIN LOG	Closes the SSL server console log and sends console log to the server owner.	“SSLADMIN LOG Command” on page 490
SSLADMIN QUERY	Displays operational information about the SSL server.	“SSLADMIN QUERY Command” on page 491
SSLADMIN REFRESH	Instructs the SSL server to update internally-maintained key database information.	“SSLADMIN REFRESH Command” on page 497
SSLADMIN RESTART	Instructs the SSL server to quiesce and IPL.	“SSLADMIN RESTART Command” on page 497
SSLADMIN SET	Establishes an SSL or TCP/IP server user ID defaults so subsequent SSLADMIN commands are directed automatically to a specific SSL server.	“SSLADMIN SET Command” on page 497
SSLADMIN START	Initializes an SSL server or servers, as determined by the SSLSERVER option, or its default or saved value.	“SSLADMIN START Command” on page 498
SSLADMIN STOP	Shuts down the SSL server.	“SSLADMIN STOP Command” on page 498
SSLADMIN SYSTEM	Instructs the SSL server to issue a specified CP or CMS command.	“SSLADMIN SYSTEM Command” on page 498
SSLADMIN TRACE / NOTRACE	Starts or stops tracing SSL server activities while the server is running.	“SSLADMIN TRACE/NOTRACE Command” on page 500

General SSLADMIN Command



Purpose

Use the SSLADMIN command to perform server administration tasks. The generalized form of this command, described here, serves to provide information about diagnostic operands that may be specified for any of the SSLADMIN commands summarized in [Table 40 on page 487](#). In addition, the return codes that pertain to the SSLADMIN command are listed. Detailed SSLADMIN command information is provided in later sections. the command descriptions.

Note: To use SSLADMIN commands, the server code disk (TCPMAINT 591) must be accessed, and the invoking user ID must be designated as an SSL server administrative user, via an appropriate DTCPARMS file :Admin_ID_List. tag definition.

Operands

diagnostic_op

*

Causes operational diagnostic messages to be produced and command data to be retained for debugging purposes.

=

Causes the supplied subcommand and operands to be forwarded to the SSL server as-is, with results displayed at the console. No validation or adjustment of operands is attempted.

Note: The previously listed diagnostic operators are intended for use in diagnosing SSL server operational problems, in consultation with the IBM support center.

command

is one of the commands listed in [Table 40 on page 487](#). For example, CLOSECON. The default is QUERY STATUS. Refer to the command descriptions in the sections that follow for detailed command information.

operands

are operands that are applicable to a given command, which modify how that command operates. Command operands can be comprised of a combination of values that you provide and command-defined keywords. Refer to the command descriptions in the sections that follow for detailed operand information.

SSLserver userid

applies this instance of the SSLADMIN command to a specific z/VM SSL server virtual machine, or all such machines. When a specific user ID is provided, it must be one that is associated with the pertinent TCP/IP server (identified via the TCPSEVER operand or its default).

SSLADMIN communicates with an SSL server (or set of servers) identified by any one of the following:

- The user ID specified with the SSLSERVER operand—a single user ID or all servers (keyword ALL).
- The SSLSERVER C environment variable (established via the command: GLOBALV SELECT CENV SETL SSLSERVER *userid*).
- All active servers, unless the subject SSLADMIN command is one that is considered potentially disruptive, as listed here:
 - RESTART
 - STOP
 - SYSTEM

In the absence of an explicit server destination (provided by one of the previous mechanisms), the above listed commands are processed **only** when one active SSL server is identified for a given TCP/IP server. If multiple such servers are detected, the subject command ends in error.

Note: To ensure consistent key database use among the servers that comprise an SSL server pool, a REFRESH command always is directed to **ALL** active servers. If an attempt is made to direct a REFRESH command to one, or subset, of the group of active servers, the subject command ends in error.

TCPserver *userid*

applies this instance of the SSLADMIN command to a specific z/VM TCP/IP stack virtual machine to allow for identification (or, confirmation) of its associated SSL server pool. The SSLADMIN command communicates with the TCP/IP stack identified by any one of the following:

- The user ID specified with the TCPSERVER operand.
- The TCPIPID C environment variable (established via the command: GLOBALV SELECT CENV SETL TCPIPID *userid*).
- The TcpiUserId value from the TCPIP DATA file.
- User TCPIP.

MONitor *seconds*

specifies the time (duration) that an SSL server or servers should be monitored for reaching a stopped state, when a RESTART command is processed. The default is 30 seconds, with minimum and maximum values of 2 and 120 seconds, respectively. If the specified value is not a multiple of the internally defined monitoring interval (2 seconds), the supplied value is rounded to the nearest such value. This option is ignored for commands other than a RESTART command.

Return Codes

0	Successful execution; no errors encountered.
1	Syntax/invocation error.
2	Internal logic error.
4	Warning(s) issued; command results may be incomplete.
8	Command processing error(s) encountered.
10	VM Interprocess communication (IPC) error encountered.
12	TCP/IP communication error encountered.

20

User is not authorized to issue command.

SSLADMIN CLEAR Command

```
➤ SSLAdmin — CLEAr — SSLserver —  
                        TCPserver —
```

Purpose

Use the SSLADMIN CLEAR command to remove a server user ID default that has been previously established by an SSLADMIN SET command.

Operands

SSLserver

indicates that any stored SSL server user ID default is to be removed.

TCPserver

indicates that any stored TCP/IP server user ID default is to be removed.

Usage Notes

- The *userid* value you supply is used to release either of the SSLSERVER or TCPIPID (GLOBALV) C environment variables, as appropriate.

SSLADMIN CLOSECON Command

```
➤ SSLAdmin — Closecon ➤
```

Purpose

Use the SSLADMIN CLOSECON command to close the SSL server console log and send it to the server owner, as identified by the :Owner. tag in a DTCPARMS file.

SSLADMIN HELP Command

```
➤ SSLAdmin — HELP ➤
```

Purpose

Use the SSLADMIN HELP command to display a list and brief explanation of SSLADMIN commands.

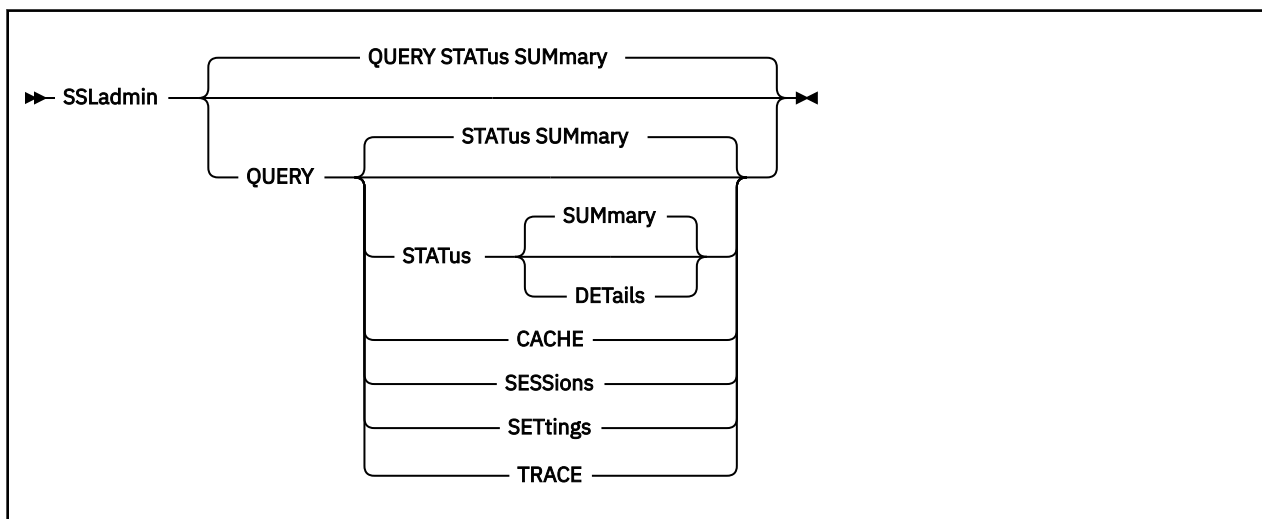
SSLADMIN LOG Command

```
➤ SSLAdmin — LOG ➤
```

Purpose

Use the SSLADMIN LOG command to close the SSL server console log and send it to the server owner, as identified by the :Owner. tag in a DTCPARMS file. This command is synonymous with the SSLADMIN CLOSECON command, and is supported to maintain compatibility with prior levels of TCP/IP for z/VM.

SSLADMIN QUERY Command



Purpose

Use the SSLADMIN QUERY command to display operational information about the SSL server, or to display current command defaults.

Operands

STATUS

requests information about SSL server status. This is the default.

SUMMARY

indicates that summarized status information is to be produced in response to a QUERY command. This is the default. Summary information is comprised of:

- an SSL server user ID
- server status (as reported by the NETSTAT CONFIG SSL command)
- the defined per-server session maximum
- the number of active sessions for a given server
- server trace status (None or Enabled)
- exempt cipher status (Boolean: Yes or No)
- cryptographic mode configuration (Boolean: Yes or No)

DETAILS

indicates that detailed status information is to be produced in response to a QUERY command. Detail information is comprised of that described for the SUMMARY operand, which then is supplemented with:

- the cryptographic modes of operation enabled for use by an SSL server
- the cryptographic modes of operation disabled from use by an SSL server
- the versions of SSL or TLS enabled for use by an SSL server
- the versions of SSL or TLS disabled from use by an SSL server

SSL Server

- the cipher suites that can be used by an SSL server
- the cipher suites that are exempt from use by a server
- server-specific trace settings that reflect any TRACE operands that are in effect

CACHE

requests information about SSL server cache usage.

SESSIONS

requests information about secure sessions.

SETtings

displays current command defaults.

TRACE

requests information about SSL server trace settings.

Examples

SSLADMIN QUERY STATUS SUMMARY - The following is an example of summary information about SSL server status:

```
ssladmin query status summary (ssl all
DTCSSL2404I Sending command to server(s): SSL00001 SSL00002 SSL00003
DTCSSL2453I Bypassing inactive server(s): SSL00005 SSL00004
DTCSSL2430I Status summary:
Server      Status    Maximum  Active   Exempt   Mode(s)
Sessions   Sessions Tracing  Ciphers? Configured?
-----
SSL00001 Active    600      600      None     Y        Y
SSL00002 Active    600      600      None     Y        Y
SSL00003 Active    600      300      Enabled  Y        Y
SSL00005 Stopped   600       0        -        -        -
SSL00004 Eligible  600       0        -        -        -
-----
Maximum Session System Limit: 3000
SSL Session High-Water Mark: 1500
```

The fields of this response supply the following information:

Server

Identifies the SSL server name.

Status

Current status of the SSL server. For a description of possible status values, see the [NETSTAT Command](#) in *z/VM: TCP/IP User's Guide*.

Maximum Sessions

Maximum number of secure sessions the SSL server supports.

Active Sessions

Current number of secure sessions.

Tracing

Indicates whether tracing is active for a server.

Exempt Cipher?

Indicates whether any ciphers are restricted from use by a server.

Mode(s) Configured?

Indicates whether the SSL server is configured to operate in compliance with any specific cryptographic standard, such as FIPS 140-2 or NIST SP 800-131A.

Maximum Session System Limit

Identifies the maximum number of secure sessions that are to be supported across all SSL servers that are associated with the subject TCP/IP stack server.

SSL Session High-Water Mark

Indicates the highest number of secure connections that have ever been active at a given time.

SSLADMIN QUERY STATUS DETAILS - When you request detailed SSL server status information, the returned response is like this example:

```

ssladmin query status details (ssl all
DTCSSL2404I Sending command to server(s): SSL00001 SSL00002 SSL00003
DTCSSL2453I Bypassing inactive server(s): SSL00005 SSL00004

DTCSSL2430I Status summary:

```

Server	Status	Maximum Sessions	Active Sessions	Tracing	Exempt Ciphers?	Mode(s) Configured?
SSL00001	Active	600	600	None	Y	Y
SSL00002	Active	600	600	None	Y	Y
SSL00003	Active	600	300	Enabled	Y	Y
SSL00005	Stopped	600	0	-	-	-
SSL00004	Eligible	600	0	-	-	-

```

-----
Maximum Session System Limit: 3000
SSL Session High-Water Mark: 1500

DTCSSL2430I Cryptographic Mode details:
Server      Status:    Modes:
-----
<*ALL*>     Enabled    FIPS-140-2
<*ALL*>     Disabled   NIST-800-131A
SSL00005    <*Data Not Available*>
SSL00004    <*Data Not Available*>

DTCSSL2430I Protocol details:
Server      Status:    Protocols:
-----
<*ALL*>     Enabled    TLSV1_1 TLSV1_0
<*ALL*>     Disabled   TLSV1_2 SSLV3 SSLV2
SSL00005    <*Data Not Available*>
SSL00004    <*Data Not Available*>

DTCSSL2430I Cipher details:
Server      State      Ciphers
-----
<*ALL*>     Included   RSA_AES_128_GCM_SHA256 RSA_AES_256_GCM_SHA384
<*ALL*>     Included   DHE_RSA_AES_128_GCM_SHA256
<*ALL*>     Included   DHE_RSA_AES_256_GCM_SHA384
<*ALL*>     Included   DHE_DSS_AES_128_GCM_SHA256
<*ALL*>     Included   DHE_DSS_AES_256_GCM_SHA384 RSA_AES_128_SHA256
<*ALL*>     Included   RSA_AES_256_SHA256 DHE_DSS_AES_128_SHA256
<*ALL*>     Included   DHE_RSA_AES_128_SHA256 DHE_DSS_AES_256_SHA256
<*ALL*>     Included   DHE_RSA_AES_256_SHA256 RSA_AES_256 DHE_DSS_AES_256
<*ALL*>     Included   DHE_RSA_AES_256 RSA_AES_128 DHE_DSS_AES_128
<*ALL*>     Included   DHE_RSA_AES_128 3DES_168_SHA DES_56_SHA
<*ALL*>     Included   DHE_RSA_DES DHE_DSS_DES
<*ALL*>     Exempt     ECDH_ECDSA_AES_256_SHA
<*ALL*>     Exempt     DH_RSA_AES_128_GCM_SHA256
<*ALL*>     Exempt     DH_RSA_AES_256_GCM_SHA384
<*ALL*>     Exempt     DH_DSS_AES_128_GCM_SHA256
<*ALL*>     Exempt     DH_DSS_AES_256_GCM_SHA384 DH_DSS_AES_128_SHA256
<*ALL*>     Exempt     DH_RSA_AES_128_SHA256 DH_DSS_AES_256_SHA256
<*ALL*>     Exempt     DH_RSA_AES_256_SHA256 RC4_128_SHA RC4_128_MD5
<*ALL*>     Exempt     DH_DSS_AES_256 DH_RSA_AES_256 DH_DSS_AES_128
<*ALL*>     Exempt     DH_RSA_AES_128 RC2_128_MD5 DHE_RSA_3DES
<*ALL*>     Exempt     DHE_DSS_3DES DH_RSA_3DES DH_DSS_3DES DH_RSA_DES
<*ALL*>     Exempt     DH_DSS_DES RC4_40_MD5 RC2_40_MD5 NULL_SHA256
<*ALL*>     Exempt     NULL_SHA NULL_MD5 NULL
DTCSSL2430I Trace settings:

```

```

Server      Normal Flow Connections Data Address      Connection
-----
S1600001 OFF      OFF  OFF      OFF  0.0.0.0..0 0
S1600002 OFF      OFF  OFF      OFF  0.0.0.0..0 0
S1600003 OFF      OFF  ON       40   0.0.0.0..0 0
S1600005 < *Data Not Available* >
SSL00004 < *Data Not Available* >

Certificate Revocation Settings:
Server      Setting      Value
-----
(*ALL*)     OCSP         ON
(*ALL*)     CDP          OFF

```

The fields of the "Status summary" portion of this response are the same as those previously described for the SSLADMIN QUERY STATUS SUMMARY example.

The fields of the "Cryptographic Mode details" portion of this response supply the following information:

- Server**
Identifies an SSL server name, or is the value <*ALL*>, which represents all SSL servers.
- Status**
Indicates whether listed cryptographic modes are enabled for, or disabled from, use by an SSL server.
- Modes**
One or more cryptographic modes of operation, such as FIPS 140-2 or NIST SP 800-131A.

The fields of the "Protocol details" portion of this response supply the following information:

- Server**
Identifies an SSL server name, or is the value <*ALL*>, which represents all SSL servers.
- Status**
Indicates whether listed versions of SSL or TLS are enabled for, or disabled from, use by an SSL server.
- Protocols**
One or more protocol versions.

The fields of the "Cipher details" portion of this response supply the following information:

- Server**
Identifies an SSL server name, or is the value <*ALL*>, which represents all SSL servers.
- State**
Indicates whether listed ciphers are included for, or excluded from, use by an SSL server.
- Ciphers**
One or more cipher suite names.

The fields of the "Trace settings" portion of this response are the same as those described later, for the SSLADMIN QUERY TRACE example.

SSLADMIN QUERY CACHE - When you request information about SSL server cache usage, the returned response is like this example:

```

ssladmin query cache (ssl all
DTCSSL2404I Sending command to server(s): SSL00001 SSL00002
                                           SSL00003 SSL00005
                                           SSL00004

DTCSSL2430I Cache details:
Server      Cache Life Cleanup Interval Cache Mode Cache Overruns New Sessions Resumed Sessions
-----
SSL00001 24:00:00 100 S 0 3 0
SSL00002 24:00:00 100 S 0 0 0
SSL00003 24:00:00 100 S 0 0 0
SSL00005 24:00:00 100 S 0 0 0
SSL00004 24:00:00 100 S 0 0 0
<*ALL*> * * * 0 3 0

```

The fields of this response supply the following information:

Server

Identifies an SSL server name, or is the value <*ALL*>, which represents all SSL servers.

Cache Life

The amount of time that an entry can exist in the cache before it expires and will be discarded, by any next cleanup action.

Cleanup Interval

The frequency (as a number of connections) at which the SSL server is to remove expired entries from the shared session cache.

Cache Mode

The caching mode in use by the server. Values that can be reported in this field are:

S

The value S (Shared) indicates that shared caching is in use by an SSL server. Normally, this value is reported by a server.

N

The value N (None) indicates that caching is not in use by an SSL server.

Cache Overruns

The number of times that addition of a new cache entry was not possible, due to the cache being full.

New Sessions

Total number of secure sessions based on new handshake agreements between the SSL server and clients (that is, not resumed from cache entries) since the SSL server was started.

Resumed Sessions

Total number of secure sessions resumed from cache entries since the SSL server was started.

SSLADMIN QUERY SESSIONS - When you request information about secure sessions, the returned response is like this example:

```
ssladmin query sessions (ssl all
DTCSSL2404I Sending command to server(s): TCPIP01
DTCSSL2430I Session information:
Server      Local Socket      Remote Socket      Type      Label      Cipher Details
-----
SSL00001 9.60.60.3..23      9.60.60.4..1031    I         TESTCERT  TLS1.2_ECDHE_ECDSA_AES_128_SHA256
(any additional SSL00001 connections would follow)
...
SSL00002 9.60.60.3..23      9.60.60.7..1036    I         TESTCERT  TLS1.2_ECDHE_ECDSA_AES_128_SHA256
(any additional SSL00002 connections would follow)
...
SSL00003 9.60.60.3..23      9.60.60.12..1045   I         TESTCERT  TLS1.2_ECDHE_ECDSA_AES_128_SHA256
(any additional SSL00003 connections would follow)
...
SSL00005 <*No Sessions*>
SSL00004 <*No Sessions*>
```

Each line of the information block (present after the field headings) represents a secure session. A secure session consists of two connections: the connection between a remote application and the SSL server, and the connection between the SSL server and a local application. The remote participant can be a client or server application, and most often is associated with a remote host (for special cases, such as a loopback connection, the "remote" application can be on the same host as the SSL server). The local application is a client or server application on the local host, for which data is protected by the secure session.

The fields of the information block supply the following information:

Server

Identifies the SSL server name.

Local Socket

The IP address and port of the local application that is a participant in the secure session.

Remote Socket

The IP address and port of the remote application that is a participant in the secure session.

Type

The type of secure connection that has been established -- Explicit (E) or Implicit (I).

Label

Label of the certificate used to authenticate the participants of the secure session. For a connection associated with a z/VM client, the string <None> appears in this field.

Cipher Details

Details (class, hash and algorithm) about the cipher used for encrypted transmissions between the session participants.

SSLADMIN QUERY SETTINGS - Assume the following SSLADMIN SET commands are issued:

```
ssladmin set tcp tcpip01
ssladmin set ssl ssl00001 ssl00003 ssl00005
```

The SSLADMIN QUERY SETTINGS command will produce the response that follows:

```
ssladmin query settings
DTCSSL2430I SSLADMIN settings:
Parameter:      Value:
-----
SSLSERVER       SSL00001 SSL00003 SSL00005
TCPSERVER       TCP0101
```

SSLADMIN QUERY TRACE - When you request information about server trace settings, the returned response is like this example:

```
ssladmin query trace (ssl all
DTCSSL2404I Sending command to server(s): SSL00001 SSL00002
                                           SSL00003
DTCSSL2453I Bypassing inactive server(s): SSL00005 SSL00004
DTCSSL2430I Trace settings:
Server      Normal Flow Connections Data Address      Connection
-----
S1600001 OFF      OFF      OFF      OFF  0.0.0.0..0 0
S1600002 OFF      OFF      OFF      OFF  0.0.0.0..0 0
S1600003 OFF      OFF      ON       ON   0.0.0.0..0 0
S1600005 <*Data Not Available*>
SSL00004 <*Data Not Available*>
```

The fields of this response supply the following information:

Server

Identifies the SSL server name.

Normal

Indicates whether NORMAL tracing is in use (ON or OFF).

Flow

Indicates whether FLOW tracing is in use (ON or OFF).

Connections

Indicates whether CONNECTIONS tracing is in use (ON or OFF).

Data

Indicates the amount of data that will be included with a CONNECTIONS trace (nn or ALL), or that DATA tracing is not in use (OFF).

Address

An IP address and port number that is to be used to limit trace activity and trace output. Zero values indicate that an entity (IP address or port number) is not in effect.

Connection

A connection number that is to be used to limit trace activity and trace output. A zero value indicate that use of a connection number is not in effect.

SSLADMIN REFRESH Command

►► SSLAdmin — REFresh ◄◄

Purpose

Use the SSLADMIN REFRESH command to instruct the SSL server to update internally-maintained key database information, so that changes to the database are employed when new connections are processed. The REFRESH command also causes the server's session cache to be cleared.

Existing connections remain unaffected by changes to the key database (such as the addition or deletion of a server certificate). However, cached session information for any such connections is lost — meaning that session keys and ciphers will need to be renegotiated for new connections established between a given client/server pair.

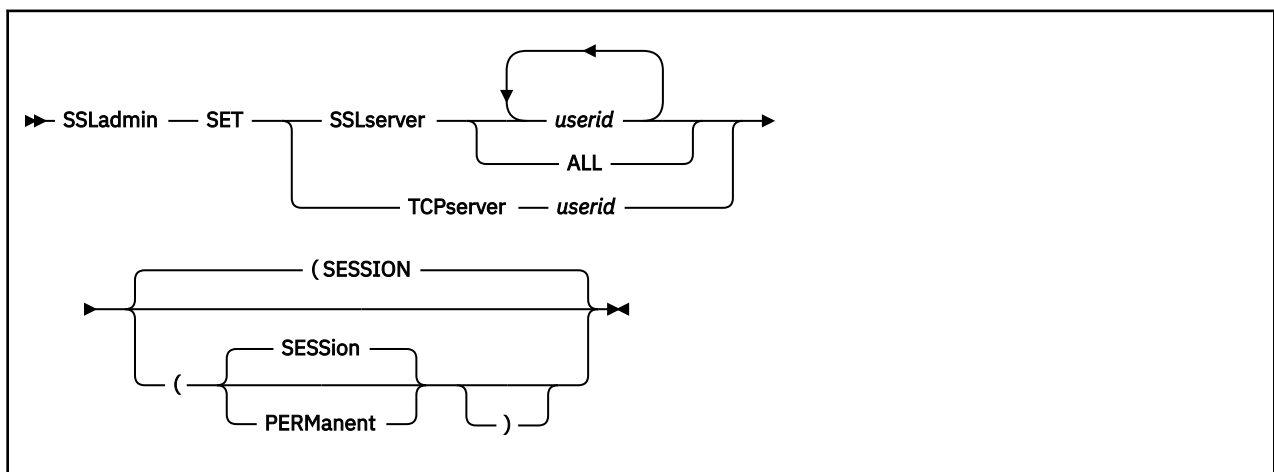
SSLADMIN RESTART Command

►► SSLAdmin — REStart ◄◄

Purpose

Use the SSLADMIN RESTART command to stop, and then restart, an SSL server in a controlled manner. For this action, SSLADMIN STOP and NETSTAT SSL START commands are used to effect the requested restart operation. The SSLADMIN command MONITOR option can be specified as part of the RESTART command, to designate the amount of time that an SSL server or servers should be monitored for reaching a stopped state, prior to again starting the affected servers.

SSLADMIN SET Command



Purpose

Use the SSLADMIN SET command to establish SSL or TCP/IP server user ID defaults so subsequent SSLADMIN commands are directed automatically to a specific SSL server (or group of such servers), when prolonged administrative actions are required.

Operands

SSLserver *userid*

establishes *userid* as the default SSL server user ID to which subsequent SSLADMIN commands are to be directed. Specify multiple user IDs so commands are directed to a limited set of active SSL servers.

To direct SSLADMIN commands to all active SSL servers that are associated with a given TCP/IP server, specify the keyword ALL.

TCPserver *userid*

establishes *userid* as the default TCP/IP server user ID with which an SSL server pool is associated.

PERManent

causes the user ID that you specify to be maintained on a permanent basis (across separate LOGONs). By default, the user ID default you specify is maintained for the duration of the current initialization (IPL) of CMS only.

SESSion

causes the user ID that you specify to be defined for only the current CMS session (generally, from LOGON to LOGOFF). This is the default.

Usage Notes

- The *userid* value you supply is used to set either of the SSLSERVER or TCPIPID (GLOBALV) C environment variables, as appropriate.

SSLADMIN START Command

```
➤➤ SSLadmin — START ➤➤
```

Purpose

Use the SSLADMIN START command to initialize an SSL server or servers, as determined by the SSLSERVER option, or its default or saved value.

SSLADMIN STOP Command

```
➤➤ SSLadmin — STOP ➤➤
```

Purpose

Use the SSLADMIN STOP command to shut down the SSL server.

SSLADMIN SYSTEM Command

```
➤➤ SSLadmin — SYStem — command ➤➤
```

Purpose

Use the SSLADMIN SYSTEM command to instruct the SSL server to issue a specified CP or CMS command.

Note: The SSLADMIN SYSTEM command is intended for use in diagnosing SSL server operational problems, in consultation with the IBM support center.

Operands

command

The CP or CMS command to be issued within the SSL server. The results of this command are displayed at the server console only; the return code from the command is reported in an SSLADMIN command response message.

Usage Notes

1. By default, only CMS command MODULES are supported; EXEC and CP commands are not directly supported. For this reason, CP commands must be prefixed with "CP", where as commands that are implemented using REXX or EXEC2 must be prefixed with "EXEC". If such a command is not prefixed as described, the SSLADMIN SYSTEM command will report a command return code of **-3**.

Examples:

To invoke the CP command QUERY TIME, use the command:

```
ssladmin system cp query time
```

To use the CMS OPENVM command (implemented using REXX) to query the file systems mounted by the SSL server, use the command:

```
ssladmin system exec openvm query mount
```

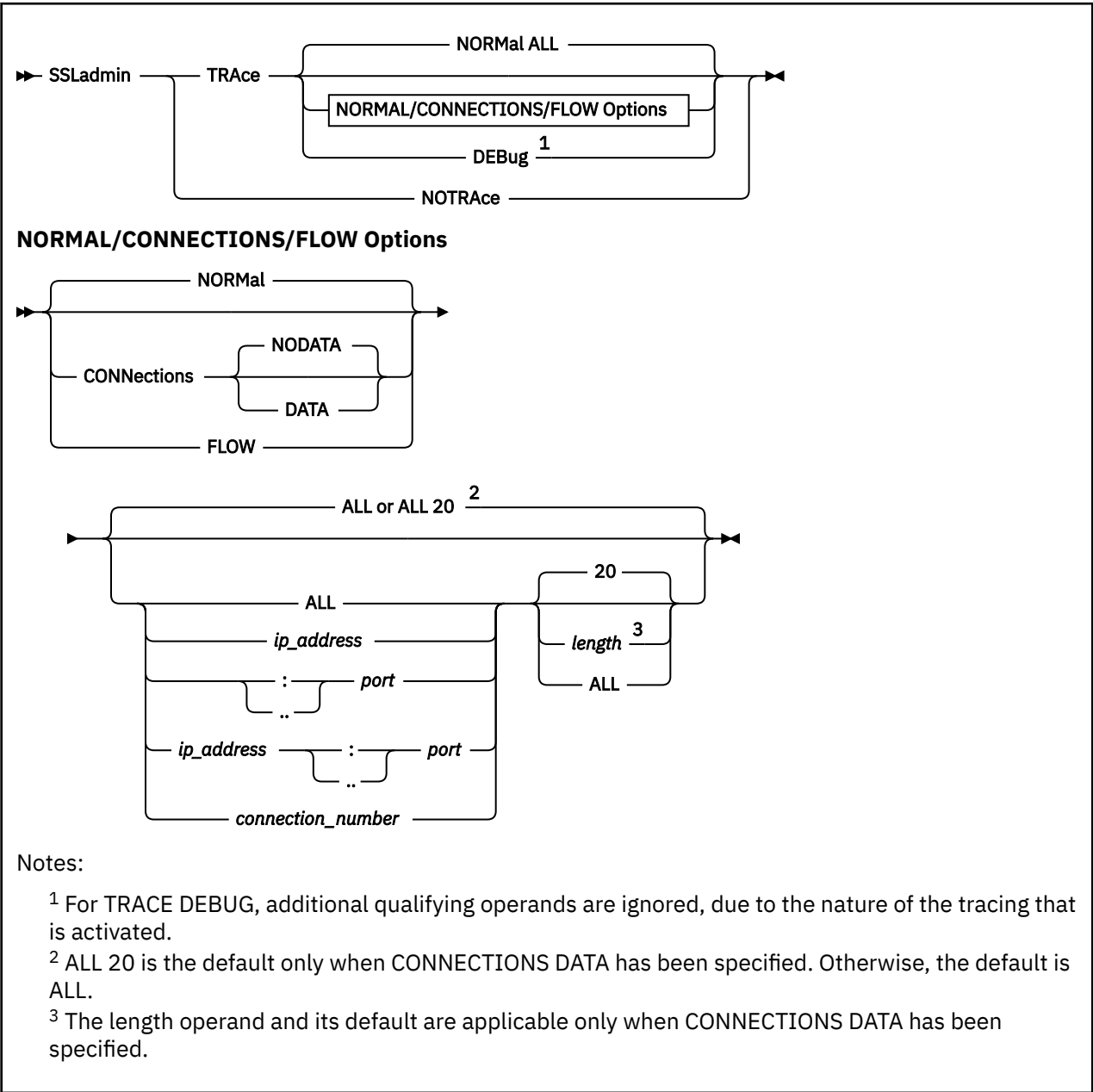
2. To process a command that requires parentheses, and at the same time employ SSLADMIN command options, enclose the supplied system command within any pairing of these characters -- / (slash) \ (backslash) ' (single quote) or " (double quote) -- such that the selected character is not part of the subject command string.

Example:

```
ssladmin system 'listfile ssl* module * ( date' (tcp tcpiptst
```

3. Care should be taken so that the commands or programs invoked within the SSL server using the SSLADMIN SYSTEM command do not adversely affect its operation. One should avoid the use of TCP/IP-oriented commands, as well as commands that can cause CMS storage management changes, extended wait conditions (perhaps due to the need for user interaction), or that otherwise adversely affect server performance. Without such care, unpredictable results or other operational errors can occur.

SSLADMIN TRACE/NOTRACE Command



Purpose

Use the **SSLADMIN TRACE/NOTRACE** command to start or stop tracing SSL server activities while the server is running.

Operands

TRACE

specifies that tracing is to be performed.

NORMAL

specifies that a trace entry is recorded to indicate a successful connection. This is the default if **TRACE** is specified.

CONNECTIONS

specifies that a trace entry is recorded for connection state changes and handshake results.

NODATA

specifies that no data is displayed for send and receive trace entries. This is the default if CONNECTIONS is specified.

DATA

specifies that data is to be displayed for send and receive trace entries, when CONNECTIONS is specified. By default, 20 bytes of data are presented in hexadecimal, as well as in ASCII and EBCDIC, in unencrypted form. The *length* operand can be used in conjunction with this operand to cause more or less data to be presented when CONNECTION activity is traced.

FLOW

specifies that flow of control and system activity are traced.

DEBug

specifies that extensive tracing is done for all control and system activity as well as data on all connections. Note that additional operands are ignored.

ALL

specifies that tracing is done for all connections. This is the default if TRACE is specified.

ip_address

specifies that tracing is done only for activity associated with this IP address.

:port***..port***

specifies that tracing is done only for activity associated with this port. The port number must be specified as a number in the range of 1 to 65535. The omission of a port value indicates that all available ports are to be traced.

Note: The format *:port* is not valid with IPv6 addresses, use *..port* instead.

connection_number

specifies that tracing is done only for activity associated with this connection number.

length

specifies the number of bytes of data to be presented in unencrypted form, when the CONNECTIONS DATA operand is used. The length must be specified as 0, or as a number in the range of 1 to 65535. The value zero (0) or the keyword ALL indicates that all available data is to be presented. The default is to display 20 bytes of data. Note that a suitable tracing target (such as an IP address, port, or connection number) must be designated when a length value other than the default is to be used.

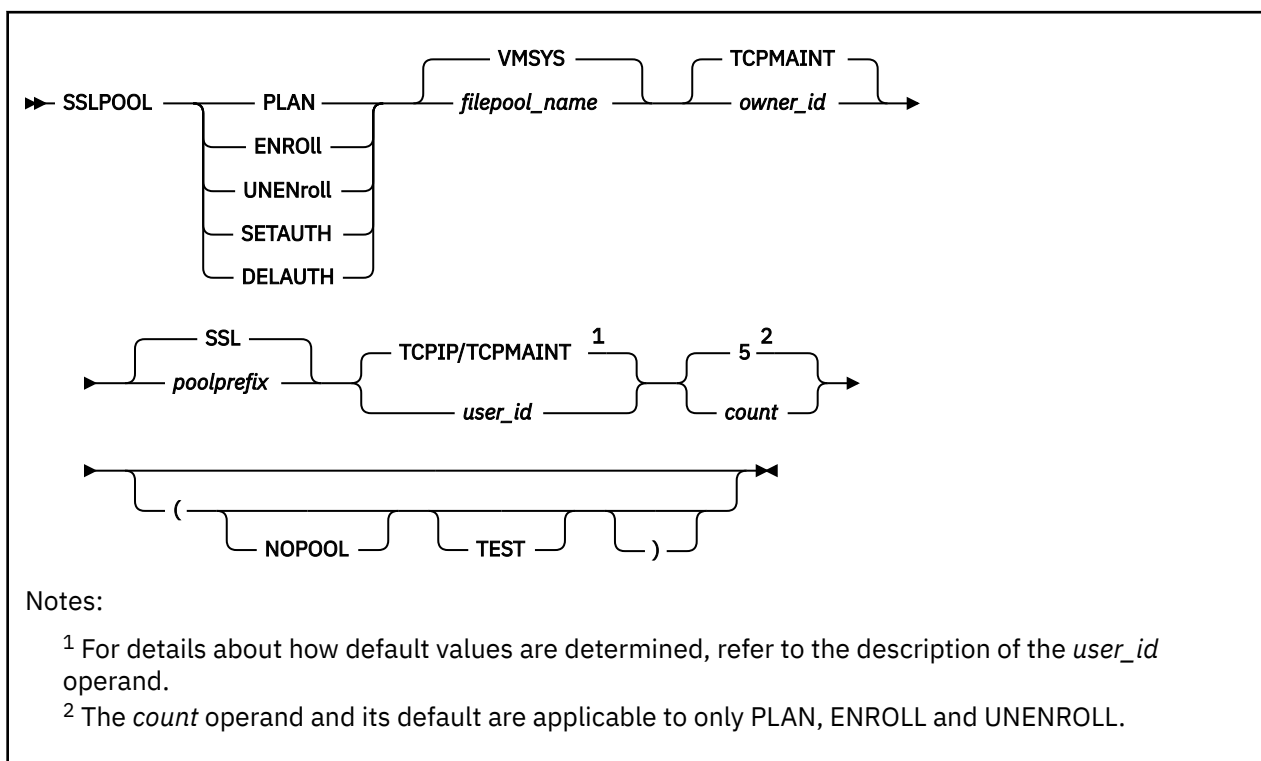
NOTRACE

specifies that all tracing is turned off.

Usage Notes

- For information about trace output, see the *z/VM: TCP/IP Diagnosis Guide*.
- This command turns tracing on or off for the current SSL server session only. If TRACE is specified as a :Parms. tag startup parameter in your DTCPARMS file, and you issue SSLADMIN NOTRACE to turn tracing off, tracing begins again each time the SSL server is restarted.

SSLPOOL Command



Purpose

Use the **SSLPOOL** command to generate planning information (comprised of sample CP directory definitions and sample DTCPARMS file entries) that can be used to assist with defining a "pool" of SSL server machines for a given TCP/IP stack virtual machine.

SSLPOOL also can be used to enroll the subject server machines in a designated Shared File System (SFS) file pool, and establish files and authorizations to facilitate their use. Similar processing, to establish appropriate authorizations for SSL server pool administrative user IDs, can be performed with this command as well. For such actions, the **SSLPOOL** command should be run by the TCP/IP installation and service user ID (for example, **6VMTCP20**), which by default, has the file pool administrative authority necessary to effect the required changes.

With each invocation, the **SSLPOOL** command updates a log file (**SSLPOOL \$MSGLOG**) with information that pertains to the command action and parameters used, as well as messages (informational or otherwise) that result from processing these values. The **\$MSGLOG** file is formatted such that it can be reviewed using the **VMSES/E VMFVIEW** command, with results for the most-recent **SSLPOOL** invocation placed at the beginning of the file.

When the **TEST** option is used, commands that would have been issued by **SSLPOOL** without the use of this option also are logged. Potential commands are logged in this manner so one can determine what actions **SSLPOOL** will take prior to its use to effect any changes.

Operands

PLAN

indicates the supplied operands are to be used to generate planning information (stored in the file **SSLPOOL PLANINFO**) for the purpose of defining and configuring an SSL server user ID pool.

Information saved in the **SSLPOOL PLANINFO** file is comprised of:

- a CP directory **USER** entry for defining the SSL server pool

- a CP directory user PROFILE entry for defining the SSL server pool
- a CP directory USER entry for defining the SSL Discontiguous Saved Segment (DCSS) Management agent server
- DTCPARMS file definitions, to account for the selected SSL server configuration (a server pool or a non-pool, single-instance server), the SSL DCSS Management agent server, and TCP/IP (stack) server updates
- TCP/IP server (PROFILE TCPIP, or equivalent) configuration file statements.

ENROLL

indicates that Shared File System (SFS) directories, server PROFILE EXEC, and SFS file aliasing necessary for running the servers that comprise an SSL server pool are to be created.

A common-use, "work space" SFS directory is created for the SSL server pool, in the file space of the designated owning user ID.

UNENroll

indicates that the SFS directories, server PROFILE EXEC, and SFS file aliasing previously established for an SSL server pool are to be removed.

SETAUTH

indicates that the SFS authorizations are to be established for the designated user ID, to facilitate administration of the server pool.

DELAUTH

indicates that previously established SFS authorizations are to be removed for the designated user ID.

filepool_name

the name of the SFS file pool in which pool servers are to be (or, are already) enrolled. The default is **VMSYS**.

owner_id

the user ID that is to own the SFS filespace that is to be used as common workspace by the SSL server pool. The default is **TCPMAINT**.

poolprefix

the prefix used for the SSL server user ID pool. This value must match that specified with the CP directory USER statement that defines the server pool. The default is SSL.

user_id

when either of the PLAN, ENROLL or UNENROLL operands has been specified, this is the user ID of the TCP/IP stack with which the SSL server pool is associated. For these operands, the user ID value is obtained (in order) from the following:

- the user ID specified with SSLPOOL command
- the TCPIPID C environment variable (as established via the command: **GLOBALV SELECT CENV SETL TCPIPID** userid)
- the TcpiUserId value from the TCPIP DATA file
- user ID **TCPIP**

For the SETAUTH and DELAUTH operands, this is the user ID for which SFS authorizations are to be established or removed, to facilitate or negate administration of the server pool. For these operands, the default is **TCPMAINT**.

count

the number of servers that comprise the SSL server pool. This value must correspond to the number of pool servers determined by the LOW and HIGH operands of the CP directory USER statement that defines the server pool. The default is **5**. This operand and its default are applicable to only the PLAN, ENROLL and UNENROLL operands.

The *count* value also is used by SSLPOOL to determine values for the SSLLIMITS statement that is included in planning information. The MAXSESSIONS value for this statement is calculated by multiplying the supplied *count* value with an appropriate MAXPERSSLSERVER value. The

MAXPERSSLSERVER value used is itself based on the count value as, summarized in [Table 41](#) on page 504:

<i>Table 41. SSLPOOL Command MAXPERSSLSERVER Values</i>	
count	MAXPERSSLSERVER Value
1	100
2-4	120
5 or greater	600

The MAXPERSSLSERVER values that are applied in this manner are recommended values, based on IBM performance analysis results. Note that these values can be further adjusted to meet requirements for your installation.

Options

NOPOOL

instructs SSLPOOL to handle the POOLPREFIX operand as a user ID instead of as a server pool prefix value, and to create planning information appropriate for a non-pool, single-instance SSL server. For this option, a server *count* value of **1** becomes effective, which overrides any such value supplied with the SSLPOOL command.

If this option is specified with other than the PLAN operand, an error results.

TEST

inhibits the execution of commands that can temporarily or permanently alter resources or the operating environment. Such commands are cited via console messages, rather than executed to allow the potential for changes to be evaluated prior to effecting those changes. Non-disruptive commands (such as query commands) still are issued with results acted upon, as warranted.

Usage Notes

- Operands are positional and must be specified in order.
- A single period (.) can be specified for a given operand, to indicate that its default should be used. Specifying operands in this manner might be useful in cases when only a few operands need to be specified with other than their default values.

For example, the command that follows will create planning information for an ST1 server pool for the TCPIPT1 TCP/IP server, using the file pool, owner and count defaults:

```
sslpool plan . . st1 tcpipt1 .
```

- By convention, the common-use "work space" SFS directory created during processing of the ENROLL command operand is named: **SSLPOOL_poolprefix**

Thus, for the *filepool_name*, *owner_id* and *poolprefix* operand defaults, the work space directory created is: VMSYS:TCPMAINT.SSLPOOL_SSL

This naming convention also is presumed when commands that employ either of the UNENROLL, SETAUTH, or DELAUTH operands are processed.

- For SSLPOOL commands that employ the ENROLL, UNENROLL, SETAUTH and DELAUTH command operands, SFS file pool administrative authority for the subject file pools is required.

Migrating Certificates From a Prior-Level SSL Server Certificate Database

For detailed information about migrating certificates from an SSL server certificate database (established for an SSL server implementation provided with prior levels of TCP/IP for z/VM), see *IBM: TCP/IP for*

z/VM Secure Socket Layer (SSL) Server Configuration Information and Requirements at TCP/IP for z/VM
(<https://www.ibm.com/vm/related/tcpip>).

Chapter 16. Configuring the TCP/IP Server

The TCPIP virtual machine provides the primary TCP/IP service called the *stack*. The stack supports the application programming interfaces and controls the network interfaces.

This topic and those that follow describe the statements and commands used to configure the TCP/IP stack. They also explain how you can change the configuration dynamically using the IFCONFIG or OBEYFILE command, as well as how to start and stop the TCP/IP stack. For more information about hardware and system requirements that are needed before configuration, see [Chapter 1, “Planning Considerations,”](#) on page 1.

To enable TCP/IP services, you must perform the following configuration steps.

Configuration Steps

1. Configure the TCPIP virtual machine.
2. Define client system parameters in the TCPIP DATA file (see [Chapter 3, “Defining the TCP/IP System Parameters,”](#) on page 13).
3. Create a local site table (see [Chapter 4, “Configuring the Local Host Files,”](#) on page 27).

Note: If you used IPWIZARD to create an initial TCP/IP configuration during z/VM installation, you have already created the following configuration files:

- PROFILE TCPIP
- SYSTEM DTCPARMS
- TCPIP DATA

For information about using IPWIZARD, see *z/VM: Installation Guide*. If these files do not meet the needs of your installation, follow the configuration steps above to customize these files as needed.

Even if you ran IPWIZARD, you still need to create local site tables. See [Chapter 4, “Configuring the Local Host Files,”](#) on page 27.

TCPIP Virtual Machine Configuration Process

To configure the TCPIP virtual machine, perform the following activities:

TCPIP Virtual Machine Configuration Steps:
1. Create a Multiprocessor Configuration (Optional) 2. Update the DTCPARMS file. 3. Create an initial configuration file.

Step 1: Create a Multiprocessor Configuration

The TCPIP virtual machine can exploit a multiprocessor configuration. It does so by using processors you designate to run specific device drivers. This allows the TCPIP load to be spread across multiple real processors and, in high-activity environments, can improve responsiveness and throughput. If your TCPIP load ordinarily uses a substantial portion of a single processor, there might be benefits to creating a multiprocessor configuration. Otherwise, the uniprocessor configuration that is created for the TCPIP virtual machine during installation is adequate and you can proceed to **Step 2** below.

You must decide how many processors to add to the TCPIP virtual machine configuration. The maximum number of processors supported is seven. The number of additional processors that can be used is limited by the number of DEVICE statements in your configuration for devices of type: LCS, CTC, IUCV, PVMIUCV, HIPERS, and OSD. In general, the most benefit is gained by using additional processors for the highest activity devices. Several devices can be associated with the same processor. If no association is

defined for a device, or if the processor designated for it does not exist, it uses the base processor. The base processor is used to run all work not associated with a device driver, such as sockets processing, the Telnet server, and TCP-layer functions.

Create a multiprocessor configuration by changing the TCP/IP virtual machine User Directory entry. First, change the MACHINE statement to specify the number of virtual processors in the configuration. For example, to create a three-processor configuration, you can use a MACHINE statement similar to the following:

```
MACHINE ESA 3
```

Then, add CPU statements to define the virtual processors. Ordinarily CPU 0 is defined as the base; TCP/IP requires that processors are numbered sequentially from 0. For example, to define three virtual processors, add the following statements:

```
CPU 00  
CPU 01  
CPU 02
```

For more information about the z/VM User Directory, see [z/VM: CP Planning and Administration](#).

Finally, when you configure DEVICE statements in “[Step 3: Create an Initial Configuration File](#)” on page 508, use the CPU option to specify the virtual processor associated with a device. For example, to associate device HIPR1, a HiperSockets device, with CPU 2, use a statement similar to the following:

```
DEVICE HIPR1 CPU 2 HIPERS 1D00 PORTNAME REDOCT
```

Step 2: Update the DTCPARMS File

When the TCP/IP stack server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server.

Tags that are significant for the TCP/IP stack server are:

```
:Nick.TCPIP  
:Attach.  
:Authlog.  
:DCSS_Parms.  
:Vctc.  
:Vnic.
```

Note: Modify the DTCPARMS file for the TCP/IP stack server if you:

- require real addresses to be attached to the server during server initialization
- must alter the name of the file used for logging unauthorized attempts to use TCP/IP services
- provide secure communications support (with an SSL server pool).
- require virtual channel-to-channel devices to be defined and coupled during server initialization
- require a virtual network interface (NIC) to be defined and coupled during server initialization
- must alter selected server attributes, such as the user ID of the virtual machine that receives its console log

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

Step 3: Create an Initial Configuration File

When the TCPIP virtual machine is started, TCP/IP operation and configuration parameters are read from an initial configuration file. TCP/IP searches for an initial configuration file in the following order and uses the first file present in that order:

1. *userid* TCPIP, where *userid* is the user ID of the TCP/IP server
2. *node_name* TCPIP, where *node_name* is the system node name returned by the CMS IDENTIFY command
3. PROFILE TCPIP.

If no file is found, TCP/IP uses server default values.

To customize your system, specify system operation, Telnet, and network parameters in the configuration file by using the configuration statements listed in Table 44 on page 524. Complete statement syntax and descriptions are in alphabetical order in [“TCP/IP Configuration Statements” on page 523](#).

A sample initial configuration file is provided as PROFILE STCPIP on the TCPMAINT 591 disk. Name your configuration file according to the file naming conventions previously described, modify the file to fit your requirements, and store the file on the TCPMAINT 198 disk.

Note: You can place most TCP/IP configuration statements in a file separate from the initial configuration file and process the separate file to change the TCP/IP configuration dynamically. For more information, see [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#).

Routing Support

TCP/IP supports static and dynamic routing. Whether you use static or dynamic, the routing mechanism is the same: TCP/IP routes a packet by searching its routing table for the most specific route known.

TCP/IP supports multiple equal-cost routes. The routing table can maintain as many equal-cost paths to a particular destination as allowed by the size of the TCP/IP virtual machine. Multiple equal-cost routes can be either defined statically or added dynamically by MPRoute. MPRoute provides at most 16 equal cost routes to each destination. For more information, see the GATEWAY statement [“Usage Notes” on page 577](#) dealing with multiple equal-cost routes.

Static Routing

Use the GATEWAY statement in PROFILE TCPIP to define any static routes. These routes do not change automatically in response to network topology changes, except when ICMP redirect is enabled and the change is due to an ICMP redirect. If a destination is down or unreachable through a static route, the static route remains in the routing table, and traffic continues to be sent toward that destination without success.

Dynamic Routing

With dynamic routing, routing table entries are dynamically managed and can automatically change in response to network topology changes. For IPv4, these routes are managed by a routing daemon, MPRoute. For IPV6, routes are managed by MPRoute or learned by listening to router advertisement messages from other routers (or both).

MPRoute

MPRoute is the recommended routing daemon for TCP/IP for z/VM. MPRoute is an IP routing daemon that supports RIP Version 1, RIP Version 2, RIPng for IPv6, OSPF, and IPv6 OSPF protocols. MPRoute can be run on IPv4, IPv6 (or both) interfaces. Either OSPF or RIP (or both) protocols can be used to maintain the routing table dynamically. MPRoute can detect when a route is created, is temporarily unavailable, or if a more efficient route exists. For IPv4 interfaces, you can send either RIP Version 1 or RIP Version 2, but not both, at the same time on a single interface; however, you can configure a RIP interface to receive both versions. When configuring MPRoute, keep in mind the following:

- MPRoute makes use of its own configuration file (MPROUTE CONFIG). MPRoute does not make use of the GATEWAY configuration statement. For IPv4 interfaces, you configure the subnet mask, destination address, and other parameters with the OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements in the MPRoute configuration file. For IPv6 interfaces, you configure using the IPV6_OSPF_INTERFACE, IPV6_RIP_INTERFACE, and IPV6_INTERFACE statements.

- For IPv4 interfaces, you can configure the MTU value for an interface using the OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements in the MPRoute configuration file, or by specifying the MTU value on the MTU option on the LINK statement in PROFILE TCPIP. If you use the LINK statement MTU option, set the MTU value on the OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements in the MPROUTE CONFIG file to zero. If you do not set the value to zero, and the value does not match the MTU option on the LINK statement, an error message is displayed.

For more information, see [Chapter 8, “Configuring the MPRoute Server,”](#) on page 193.

Network vs. Host Routing

It is not recommended that you put multiple VM hosts with more than one hop count in the same subnet. This causes router confusion in selecting which path to choose to reach the destination. This confusion can be removed by advertising RIP host routes when running MPRoute. To enable host route advertising in MPRoute, configure RIP_Interface Send_Host_Routes=YES. Another solution for this configuration is to use static routes. With the additional host route information in the RIP updates, the router (9.130.3.10 in [Figure 6 on page 510](#)) uses the host routes to reach the destinations, even though only one subnet route is being assigned to one of the interfaces.

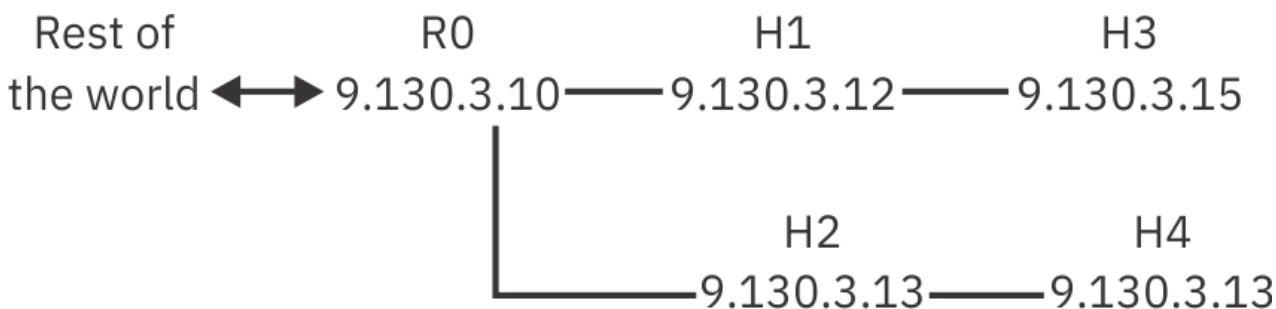


Figure 6. Host routing under single subnet

The recommended practice for configurations involving destinations that are more than one hop count away from the source is to assign different subnets for each interface. For example,

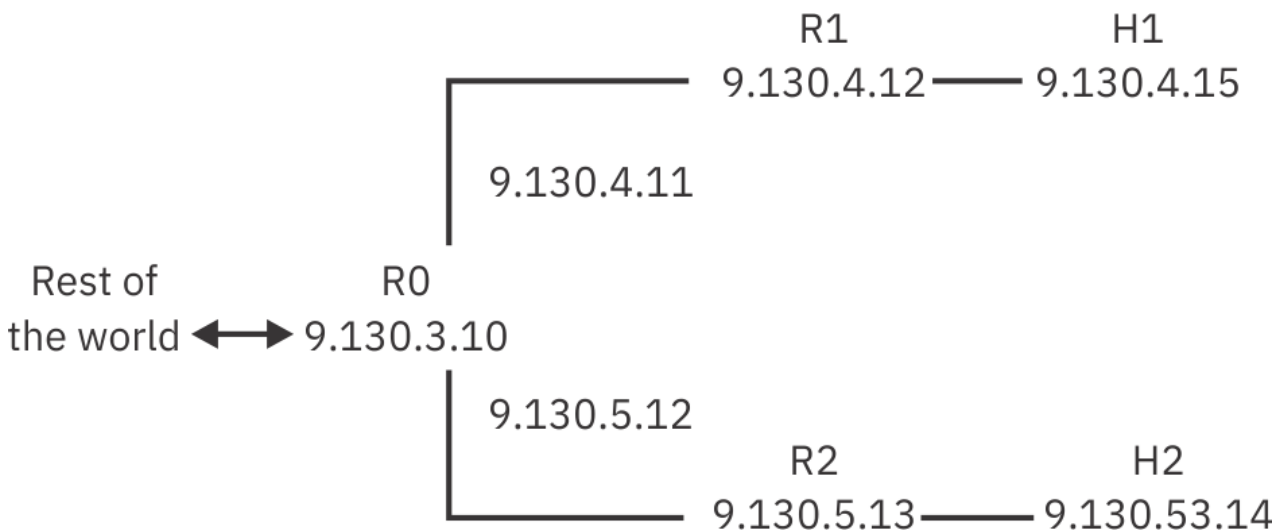


Figure 7. Subnet assignment for destinations beyond a single hop

Because VM hosts (routers R1 and R2) are intermediate, they must be assigned different subnets (or networks). Notice that we have to assign new IP addresses 9.130.4.11 and 9.130.5.12 on R0 to define 9.130.4.x and 9.130.5.x subnets. This is necessary after expanding the network to reach other hosts. That is, assume that the original configuration is:

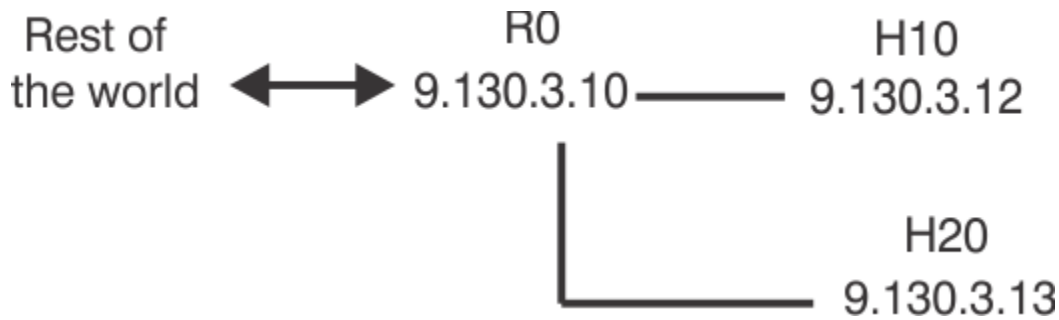


Figure 8. Basic host routing configuration

This configuration is valid because VM hosts H10 and H20 are not routers, but hosts at end points from router R0. Now, when H1 and H2 are added to the configuration, H10 and H20 are no longer hosts by definition, but are routers. Therefore, in this case, assign different subnets to each routing interface as illustrated in [Figure 7 on page 510](#).

You can add more hosts in each subnet as long as those hosts do not become routers. For example, hosts H01, H10, and H21 are added to the configuration within their subnets:

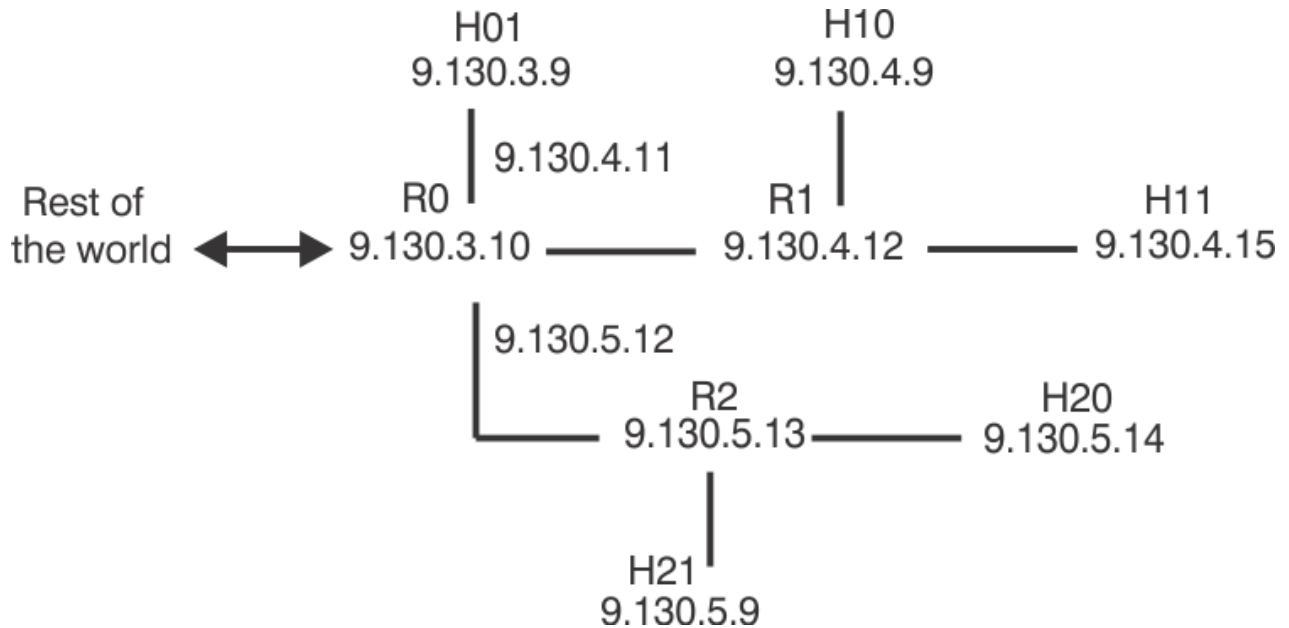


Figure 9. Adding hosts to subnetted interfaces

A good reason for using different subnets for each routing interface is that routers communicate using network-specific routes, not host-specific routes. According to the RFCs for RIP, host-specific routing is optional. For this reason, there is no guarantee that routers in the network will communicate using host routes. That is, there are routers (for example, Wellfleet, Proteon) that will ignore host routes in the RIP broadcasts and rely on network-specific routes only. IP addresses, and their subnet masks, must be used with care when defining RIP networks. In addition, advertising host routes in MPRoute must be used carefully, and the network configuration must be planned with the assurance that the adjacent routers accept the host routes.

Multicast Support

There are three methods that exist for sending data in a network environment:

1. Unicasting
2. Broadcasting
3. Multicasting

In unicasting, a datagram is sent to each host that requests it. A disadvantage of this is that the number of hosts is limited by the bandwidth on the sender. In broadcasting, copies of a message are sent to all hosts in the network, whether they request it or not. A disadvantage to this is that the hosts must be members of a single subnet.

In multicasting, a message is sent to multiple selected hosts in a group known as a multicast group. Multicasting provides the following advantages:

- Only one copy of a multicast message is sent over any link in the network; copies of a message are only made when paths diverge at a router.
- The sending server does not have to maintain a list of recipients because all hosts in a multicast group are identified by a single IP-destination address.
- Membership in a multicast group is dynamic:
 - Hosts can join or leave at any time
 - A host can be a member of more than one group at a time
 - The location of hosts within a network is unrestricted
 - The number of members in a multicast group is unlimited. When hosts join, you do not have to increase bandwidth.
- For applications that support IP multicast, a single group address can have multiple data streams on different port numbers on different sockets, in one or more applications. Multiple applications can share a single group address on a port.
- Multicasting reduces the load on the sending server, which no longer has to support multiple sequential or concurrent unicast sessions.

Note: TCP/IP contains host support of multicast, but not router support. Host support is based on implementation of the RFC 112 standard.

To implement multicasting in your applications:

- Use the C `setsockopt()` and `getsockopt()` calls to implement multicast functions. For information about these calls, see [z/VM: TCP/IP Programmer's Reference](#).
- Use the GATEWAY statement in the PROFILE TCPIP configuration file to specify static routes for multicast datagrams. At a minimum, define a default multicast route.

Multiple Interface Network Support

TCP/IP supports multiple interfaces and IP addresses on the same network, providing redundant paths to other hosts or routers on directly attached networks. When you configure multiple interfaces to the same network, one of them provides the primary path to hosts and routers on that network; the others are secondary paths.

In general, the interface used for traffic originating at your VM host is the one that provides the first active route to a destination, according to the IP routing table. If the destination route is not available, the interface that provides the first active default route is used. Furthermore, the home IP address of the selected interface is used as the local address for an application socket if SOURCEVIPA is not enabled and the socket is not bound to a specific local IP address.

Note:

1. The first primary path can be specified in the PRIMARYINTERFACE statement. On remaining interfaces, the first link for each defined network or subnetwork in the list of home addresses is the primary one.
2. If no PRIMARYINTERFACE statement is configured, the first link specified for each defined network or subnetwork in the list of home addresses is the primary one.

Interface Takeover for Local Area Networks

The TCP/IP stack in z/VM provides transparent fault-tolerance for failed (or stopped) IPv4 or IPv6 devices, when the stack is configured with redundant connectivity into a LAN. This support is provided by the z/VM TCP/IP interface-takeover function, and applies to IPv4 QDIO and LCS Ethernet devices and to the IPv6 QDIO Ethernet devices.

At device startup time, TCP/IP learns of redundant connectivity onto the LAN, and uses this information to select suitable backups in the case of a future failure of the device. This support makes use of ARP flows (for IPv4 devices) or neighbor discovery flows (for IPv6 devices), so upon failure (or stop) of a device, TCP/IP immediately notifies stations on the LAN that the original IPv4 or IPv6 address is now reachable through the backup's link-layer (MAC address). Users targeting the original IP address will see no outage due to the failure, and will be unaware that any failure occurred.

Since this support is built upon ARP or neighbor discovery flows, no dynamic routing protocol in the IP layer is required to achieve this fault tolerance. To enable this support, you must configure redundancy onto the LAN:

- You need redundant LAN adapters.
- You must configure and activate multiple LINKs onto the LAN.

Restriction: An IPv4 device cannot back up an IPv6 interface, and an IPv6 interface cannot back up an IPv4 device.

Note: If static routing is used, there needs to be a static route to the LAN subnet over each interface onto the LAN. There also needs to be a default route and routes to destinations not directly attached to the LAN over each interface.

The interface-layer fault-tolerance feature can be used with VIPA addresses, where applications can target the VIPA address, and any failure of the real LAN hardware is handled by the interface-takeover function. This differs from traditional VIPA usage, where dynamic routing protocols are required to route around real hardware failures.

Virtual IP Addressing (VIPA)

Virtual IP Addressing (VIPA) frees other hosts from depending on a particular physical network interface for communication with a z/VM TCP/IP stack. Without VIPA, other hosts are bound to one of the VM host's home IP addresses and, therefore, to a particular physical network interface (for example, a device or adapter). If that interface fails, the associated connections are terminated. VIPA provides an IP address that is associated with a z/VM TCP/IP stack but not with a specific physical network interface. This allows hosts that connect to the z/VM TCP/IP stack to send data on whatever paths are selected by the routing protocols; thus, VIPA provides tolerance of physical network interface hardware failures.

To achieve network interface independence, VIPA relies on a virtual device¹ and a virtual IP address. The virtual device is always active and never experiences a failure. A virtual IP address is the home address for a virtual device, which has no associated physical network interface. Inbound packets whose destination is a virtual IP address can be routed through any of the real physical network interfaces used by a z/VM TCP/IP stack. Failure of one physical network interface is handled by routing inbound traffic to another active physical network interface. Similarly, outbound packets can be routed around physical network interface outages, assuming an additional physical network interface provides an alternate path to the final destination.

Dynamic routing protocols can be used to manage alternate paths. In general, z/VM provides the following functions:

- Automatic and transparent recovery from device failure.

When a device fails, if there is another device that provides alternate paths to the destination, and if other hosts make connections using virtual IP addresses, then:

¹ The term *virtual device*, used in this section to describe a device supported by VIPA, is not related in any way to VM's traditional virtual device support.

- IP detects the failure, finds an alternate path for each network, and routes outbound traffic to hosts and routers on networks via alternate paths.
- The result is fault tolerance for both inbound and outbound traffic, without the need to reestablish active connections that were using the failed device.
- Automatic and transparent recovery from adapter failure.

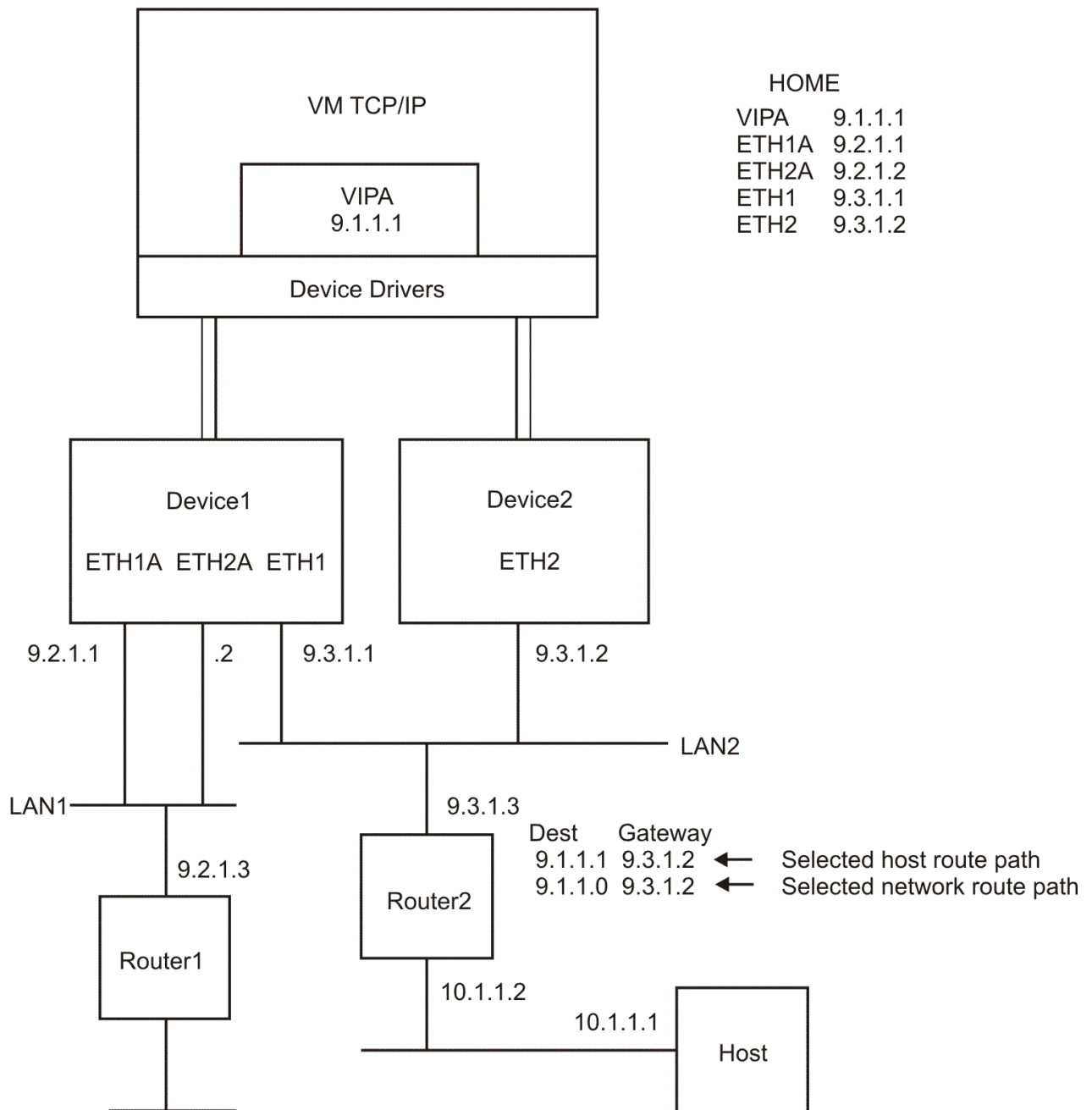
Assume that multiple physical network adapters are installed on a device. If there are multiple alternate paths defined over these adapters to the destination, and if other hosts are connecting to virtual IP addresses, then:

- IP detects the failure, finds an alternate path for each network, and routes outbound traffic to hosts and routers on those networks via alternate paths.
- The result is fault tolerance for both inbound and outbound traffic, without the need to reestablish active connections that were using the failed adapter.
- Recovery from z/VM TCP/IP stack failure (when an alternate z/VM TCP/IP stack is configured to provide the necessary redundancy).

Assume that an alternate TCP/IP stack is installed to serve as a backup and VIPA is configured on the primary TCP/IP stack. In a primary stack failure, the backup can be reconfigured to use the primary's virtual IP addresses. Client/server connections on the failed primary stack are disrupted but can be reestablished on the backup using the primary's virtual IP addresses as destinations. In addition, the temporarily reassigned virtual IP addresses can be restored to the original primary stack when recovery is complete.

Note: For requests or connections originating at a z/VM TCP/IP stack, tolerance of device and adapter failures might be achieved by using the SOURCEVIPA feature. This capability causes virtual IP addresses to be used as the source IP addresses in all outbound datagrams except those associated with routing.

[Figure 10 on page 515](#) shows an example of a VIPA configuration.



Dest Gateway
 9.1.1.1 9.2.1.1 ← Selected host route path
 9.1.1.0 9.2.1.1 ← Selected network route path

Figure 10. Single VIPA Configuration

Configuring VIPA

Assume that you want to configure TCP/IP to use a virtual IP address. The necessary steps are:

1. Add a virtual device and link to the **DEVICE** and **LINK** statements. See [“DEVICE and LINK statements for virtual IP addressing \(VIPA\) devices”](#) on page 563.
2. Add the virtual link to the **HOME** statement. See [“HOME Statement”](#) on page 579.
3. **(IPv4 only)** If the virtual IP address is to be a default local host, see [“PRIMARYINTERFACE Statement”](#) on page 606.

4. If you want tolerance of device and adapter failures for requests or connections originating at a z/VM TCP/IP stack, specify the SOURCEVIPA and/or IPV6SOURCEVIPA option in the ASSORTEDPARMS statement. For this specification to be effective, the receiving nodes in the network must be configured to recognize the virtual IP addresses, using one of the dynamic routing protocols. Otherwise, timeouts for connection or request responses will occur as a result of the virtual IP addresses not being reachable. For more information about configuring home addresses using SOURCEVIPA, see [“Configuring Source VIPA” on page 516](#).
5. For host name resolution, replace the physical IP addresses with the virtual IP address in the domain name servers.
6. Depending on routing protocols in use, your routing paths— whether they are physical network interfaces used to reach destination hosts or those used by receiving nodes to reach the virtual IP address— must be dynamically-learned and maintained. Consider the following:
 - If OSPF or RIP services are in use and Host Route Broadcasting (that is, the ability to learn host routes) is not supported by adjacent routers, the following restrictions for virtual IP addresses must be observed to gain the benefits of fault tolerance support:
 - If you use subnetting and virtual IP addresses are in the same network as the physical IP addresses, the subnetwork portions of virtual addresses must not be the same as the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the virtual IP address.
 - If subnetting is not used on any physical interface, the network portion of any virtual IP addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the virtual IP address, preferably a class C network address.
 - If OSPF or RIP services are in use and Host Route Broadcasting is supported by adjacent routers, the network or subnetwork portions of virtual IP addresses can be the same across multiple z/VM TCP/IP stacks in the network.

Note:

1. In addition to virtual links, physical links must be defined to provide the actual network attachments.
2. A virtual link cannot be defined using the GATEWAY statement; therefore, there is no "virtual" route to define and no route to display via the NETSTAT GATE command.

Configuring Source VIPA

All outbound packets sent out via an interface configured for source VIPA will have the VIPA set as the source address, unless prevented e.g. by usage of socket option SO_IGNORESOURCEVIPA. Configuring source VIPA requires that the stack has been configured for use of a virtual IP address, see [“Configuring VIPA” on page 515](#). Specifying what links should use which source VIPA address works differently in IPv4 and IPv6.

Configuring Source VIPA in IPv4

The necessary steps are:

- Add the SOURCEVIPA option to the ASSORTEDPARMS statement.
- In the HOME statement, list all interfaces that should use source VIPA after the respective VIPA's entry. For more information, see [“HOME Statement” on page 579](#).

Note:

1. Interfaces that should not use source VIPA need to be listed before the first VIPA's entry in the HOME statement.
2. If you specify SOURCEVIPA on the ASSORTEDPARMS statement, the order of addresses is important with respect to how source IP addresses are used for outbound datagrams originating at the host. In this case, TCP/IP behaves as follows:
 - In the HOME list, the virtual IP address that most closely precedes a physical IP address is used as its source IP address.

- If virtual IP addresses are coded after all physical IP addresses, no virtual addresses are used as source IP addresses.

Note: See the Examples section for more information about configuring the “HOME Statement” on [page 579](#) when SOURCEVIPA is specified.

Example

1. This example uses the SOURCEVIPA option for outbound datagrams originating at a z/VM TCP/IP stack.

Select a virtual IP address in the HOME statement as the local address. The address that most closely precedes a physical IP address is used as its local address.

Example:

```
HOME
172.2.1.1    VIPA1 ; <-- Source for LINK2 and LINK1
151.2.3.1    LINK2
151.4.1.1    LINK1
172.2.1.2    VIPA2 ; <-- Source for LINK7 and LINK6
151.2.3.2    LINK7
151.4.1.2    LINK6
```

Optionally, additional virtual IP addresses can be defined to associate a group of interfaces and serve as local addresses. VIPA1 and VIPA2 are examples of virtual links; the remaining entries are examples of actual links that are associated with physical IP addresses. Virtual IP addresses are used in outbound IP datagrams. For more information see “ASSORTEDPARMS Statement” on [page 529](#). If you specify SOURCEVIPA on the ASSORTDPARMS statement, link VIPA1 provides the virtual IP address for LINK2 and LINK1, while link VIPA2 provides it for LINK7 and LINK6.

If an outbound datagram is not to contain a source virtual IP address for a particular interface (that is, a physical IP address should always be used), the address and link entries must be suitably ordered, as shown in the following example.

```
HOME
151.4.1.1    LINK1 ; <-- No SOURCEVIPA for outbound on LINK1
172.2.1.1    VIPA  ; <-- Source for ETH1 and LINK6
151.2.3.1    ETH1
151.4.1.2    LINK6
```

Table 42 on [page 517](#) shows how various TCP/IP protocols use a virtual IP address when it is specified as a local address. "Y" indicates that the address will be used as the local address, and "N" indicates that it will not be used as the local address.

Table 42. Source VIPA Usage Chart

Destination	ICMP	TCP	RAW	UDP
Local Interface	N	N	N	N
Local network	Y	Y	Y	Y*
Remote network	Y	Y	Y	Y

Note: * Except for RIP packets

Configuring Source VIPA in IPv6

The necessary steps are:

- Add the IPV6SOURCEVIPA option to the ASSORTEDPARMS statement.
- For any IPv6 interface that should use source VIPA, you must specify the SOURCEVIPAINTERFACE option on the LINK statement for that interface. The SOURCEVIPAINTERFACE option will indicate which VIPA link the real interface should use when determining the source VIPA IP address.

Note:

1. In contrast to configuring source VIPA in IPv4, the HOME list order has no impact on the VIPA used for outbound packets.
2. For IPv6, only interfaces that have specified the SOURCEVIPAINTERFACE option on their LINK statement will use a source VIPA address. Any IPv6 interface that has not specified this option, will not use a source VIPA address.
3. If an IPv6 interface has specified a SOURCEVIPAINTERFACE link name that has multiple IP addresses associated with it, then the default source address selection algorithm (as documented in RFC 3484) will be used to determine which IP address to use as the source VIPA address.

Using VIPA to back up or restore a TCP/IP stack

Because a virtual IP address is associated with a TCP/IP stack and not with a specific physical network interface, a primary TCP/IP stack can be backed up by an alternate TCP/IP stack on the same or another z/VM system. This allows hosts that are connected to the primary TCP/IP stack to re-establish connections with an alternate stack when the primary is unavailable by using the virtual IP address of the primary. After the primary TCP/IP stack has been recovered, the temporarily reassigned virtual IP address can then be reclaimed from the alternate stack.

Whenever you back up or restore a z/VM TCP/IP stack, always consider the following:

- All sessions with servers on the failing host will be disrupted.
- Clients can use any ephemeral port number when connections are reestablished to backup servers.
- Having different port numbers for the alternate and primary servers is not recommended. If the alternate server has a different port number than the primary (for example, port 101 rather than port 21 for FTP), the client must know to use a different port (for example, 101 rather than 21). Using different port numbers does work, but can cause administrative problems.

OSA-Express adapter support

TCP/IP fully supports the OSA-Express adapters. The OSA-Express adapter provides integrated native-systems connectivity to local area networks. TCP/IP provides two levels of support for the OSA-Express adapter:

1. Queued Direct I/O Hardware Facility (QDIO) - (CHPID type OSD)
 - TCP/IP supports the following protocols:
 - For OSD devices:
 - Gigabit Ethernet
 - 10 Gigabit Ethernet
 - 25 Gigabit Ethernet
 - 1000Base-T
2. Legacy OSA LAN Emulation – (CHPID type OSE)
 - TCP/IP support is provided through the traditional LCS device support. Only 1000-BaseT models can be configured in non-QDIO mode.

Because the QDIO hardware facility allows for a direct connection between the TCP/IP stack and the OSA-Express adapter resulting in improved performance, it is highly recommended that you configure the adapter as an OSD device.

Multiple guests can exchange data directly with a single OSA-Express adapter using QDIO when they couple to a virtual switch. With a virtual switch, the traditional OSA-Express adapter function is shared by a TCP/IP for z/VM stack and the Control Program (CP) for a virtual switch. The TCP/IP stack manages the control and data connections to the device, while CP handles the data transfers. This allows CP to pass data directly to the target destination without the overhead of going through a separate TCP/IP stack acting as a router.

CP and the controlling TCP/IP for z/VM stack communicate using a private CP System Service, *VSWITCH. This allows them to share device status and other information.

Network Express adapter support

TCP/IP fully supports the Network Express adapters. The Network Express adapter provides integrated native-systems connectivity to local area networks. Network Express adapters run an enhanced version of the Queued Direct I/O Hardware Facility (EQDIO) and are configured as CHPID type OSH. The EQDIO hardware facility allows for direct memory access between the TCP/IP stack and the Network Express adapter, resulting in improved performance.

HiperSockets support

TCP/IP supports HiperSockets, an extension to the Queued Direct I/O (QDIO) Hardware Facility, providing a microcode only, low latency communications vehicle for Internet Protocol (IP) inter-program communications (IPC). Communications between programs is accomplished with traditional TCP/IP socket connections. With the use of HiperSockets, a program will not only have the ability to directly communicate with a program running within the same logical partition (LPAR), but also across any logical partition within the same Central Electronics Complex (CEC).

Secure communication support

TCP/IP can be configured to provide secure communication between remote clients and z/VM TCP/IP application servers that provide support for secure communication, as well as for z/VM TCP/IP Telnet and FTP clients. Secure communication support is provided through conjunctive processing by the TCP/IP stack server and an SSL server pool, as well as a corresponding SSL DCSS management agent server (SSLDCSSM).

Attaching network devices to the TCPIP virtual machine

During the installation process, you must make sure that network devices are attached to the TCPIP virtual machine. You can accomplish this in one of two ways:

1. Modifying the DTCPARMS file, enabling the necessary devices to be attached by using the :Attach . tag (for an example see [“Customizing Servers”](#) on page 43), or,
2. Adding the appropriate DEDICATE control statements to the TCPIP virtual machine's directory entry.

Note: A TCP/IP device address can be any hexadecimal value between 0001 and FFFF; a device address of 0000 is not valid.

TCP/IP for z/VM allows a single TCPIP virtual machine to drive multiple instances of a supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration file.

Unless otherwise noted, z/VM does not require a device definition in the system configuration file or HCPRIO. The actual device attributes are determined dynamically during device initialization. For more information about when and how devices are defined for z/VM, see [z/VM: CP Planning and Administration](#).

The DEVICE statement identifies a device (control unit, communications adapter, or software service) that provides a connection to one or more networks or to a directly-connected host. Each connection to a particular network or host is called a *network interface*. The LINK statement identifies a specific network interface in a specific device, allowing TCP/IP to use it.

The HOME statement is used to assign a network address to the interface. The START and STOP statements are used to activate and deactivate the network interface. Each LINK statement should have corresponding HOME and START statements.

Devices are not automatically started when TCP/IP initializes, so you must either include START statements in the initial configuration file or start the devices manually using the OBEYFILE command.

There are DEVICE and LINK statements for configuring the following device types:

Device type	Link to DEVICE and LINK information
Channel-to-channel adapters	“DEVICE and LINK statements for CTC devices” on page 539
HiperSockets connections	“DEVICE and LINK statements for HiperSockets connections” on page 542
Local IUCV connections	“DEVICE and LINK statements for local IUCV connections” on page 545
Remote IUCV connections	“DEVICE and LINK statements for remote IUCV connections” on page 548
LAN Channel Station	“DEVICE and LINK statements for LCS devices” on page 550
OSA-Express adapters	“DEVICE and LINK statements for OSD devices” on page 553
Network Express adapters	“DEVICE and LINK statements for OSH devices” on page 559
Virtual IP addressing (VIPA) devices	“DEVICE and LINK statements for virtual IP addressing (VIPA) devices” on page 563

Notes:

- You can use DEVICE and LINK statements to connect two TCP/IP virtual machines. See [“DEVICE and LINK statements for local IUCV connections” on page 545](#).
- You can use the OBEYFILE command to add new DEVICE and LINK statements. You cannot modify any existing statements.

When you add new LINK statements, all of the entries defined by the GATEWAY, HOME, and TRANSLATE statements are deleted. When using the OBEYFILE command to add new LINK statements, make sure you include the complete GATEWAY, HOME, and TRANSLATE statements.

For more information about OBEYFILE, see [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#).

- You can use OSD or OSH statements to configure a TCP/IP EQDIO connection.

Point-to-point connections to other hosts

A point-to-point connection is a network that consists of exactly two hosts. As with traditional networks, each host must assign a network address to its network interface. The address assigned does not need to be unique as long as the host has some other network connection for which a unique address has been assigned and you are not running MPRoute. In this case, the point-to-point link can be considered an "extension" of another.

For example, consider the following scenario:

- Hosts A and B are connected by a CTC link
- Host A is also connected to an ethernet LAN whose address is 193.1.1
- Host B is also connected to an ethernet LAN whose address is 193.1.2
- Host A's home address on its ethernet LAN is 193.1.1.1
- Host B's home address on its ethernet LAN is 193.1.2.1

Host A's configuration file could contain:

```
home
  193.1.1.1   eth1
  193.1.1.1   CTC1A

gateway
; Network      Subnet mask   First hop   Link      Packet Size
  193.1.1      255.255.255.0   =          eth1      2000
  193.1.2      255.255.255.0   =          CTC1A     2000
```

Host B's configuration file could contain:

```
home
  193.1.2.1   eth1
  193.1.2.1   CTC1B

gateway
; Network      Subnet mask   First hop   Link      Packet Size
  193.1.2      255.255.255.0   =          eth1      2000
  193.1.1      255.255.255.0   =          CTC1B     2000
```

Figure 11. Point-to-Point Link

The CTC links do not have their own home addresses. Hosts A and B are addressed by their Ethernet LAN addresses, even if the packets reach them through the CTC link.

If host B had no other network attached to it you would have to assign a separate (sub)network number to the CTC link. Even in this case, Host A does not need a separate home address for its side of the link because it can be addressed by its Ethernet LAN home address. Host B's only home address is the home address for the CTC link.

Free Pool Statements

Each control block or data buffer needed by TCP/IP is allocated from the *free pool*, the size of which is determined by the virtual storage size of the TCPIP virtual machine. The free pool is subdivided into separate pools for each type of control block or buffer. The initial size of each pool is determined by a pool configuration statement. All pools are created when TCP/IP is started, so if there is not enough virtual storage to contain their initial sizes, TCP/IP will issue messages indicating what was actually allocated and how much more virtual storage needs to be defined.

As shown in [Table 43 on page 522](#), most pools have an associated *permit size*, or threshold, which is computed as a percentage of the pool size. When the number of elements (control blocks or data buffers) remaining in a particular pool drops below the permit size, TCP/IP will send a message to every user in the INFORM list. The message is sent only once per pool until the NETSTAT RESETPOOL command is issued. [Table 43 on page 522](#) also shows the *limit size* associated with each pool. This value is either a percentage of the pool size or an absolute value and is smaller than the permit size. When the number of elements left in a pool drops to its limit size, TCP/IP attempts to allocate more pool elements dynamically and sends a message to every user in the INFORM list to report the situation and indicate whether it was alleviated.

Note: The Fixed Page Storage pool is allocated on an as-needed basis. The initial number of elements to be allocated for this pool can be specified on the FIXEDPAGESTORAGEPOOL statement, while the total number allocated is the lesser of a limiting value (also specified on this configuration statement) or the number of elements that can be allocated from virtual storage.

Messages that concern Fixed Page Storage pool allocation are issued to users in the INFORM list when the number of elements in use exceeds 90% of the upper bound of the total allocated (as previously described). This is in contrast to messages for other pools, which are issued when the number of free elements drops below 10% of the total number allocated for a given pool.

The NETSTAT POOLSIZE command displays the total number of elements, the number of elements in use, the minimum number of elements available since TCP/IP was started, and the permit size of each pool. Use this command to monitor and adjust pool sizes to ensure availability of TCP/IP services.

Table 43. *Free Pool Configuration Statements.* Each pool configuration statement is shown. The inform threshold is the percentage used to calculate the permit size. The limit is either the percentage of the pool size used to calculate the limit size or an absolute number of pool elements.

Statement	Inform Threshold	Limit	Location
ACBPOOLSIZE	10%	5%	“ACBPOOLSIZE Statement” on page 527
ADDRESSTRANSATIONPOOLSIZE	0.33%	0%	“ADDRESSTRANSATIONPOOLSIZE Statement” on page 528
CCBPOOLSIZE	10%	5%	“CCBPOOLSIZE Statement” on page 536
DATABUFFERPOOLSIZE	10%	5%	“DATABUFFERPOOLSIZE Statement” on page 537
ENVELOPEPOOLSIZE	10%	5%	“ENVELOPEPOOLSIZE Statement” on page 564
FIXEDPAGESTORAGEPOOL	10%	0%	“FIXEDPAGESTORAGEPOOL” on page 566
FOREIGNIPPOOLSIZE	10%	5%	“FOREIGNIPPOOLSIZE Statement” on page 568
IPROUTEPOOLSIZE	2%	0%	“IPROUTEPOOLSIZE Statement” on page 590
LARGEENVELOPEPOOLSIZE	10%	5%	“LARGEENVELOPEPOOLSIZE Statement” on page 591
NCBPOOLSIZE	0.33%	0%	“NCBPOOLSIZE Statement” on page 595
RCBPOOLSIZE	10%	5%	“RCBPOOLSIZE Statement” on page 608
SCBPOOLSIZE	10%	5%	“SCBPOOLSIZE Statement” on page 613
SKCBPOOLSIZE	10%	5%	“SKCBPOOLSIZE Statement” on page 614
SMALLDATABUFFERPOOLSIZE	10%	5%	“SMALLDATABUFFERPOOLSIZE Statement” on page 615
TCBPOOLSIZE	10%	5%	“TCBPOOLSIZE Statement” on page 620
TINYDATABUFFERPOOLSIZE	10%	5%	“TINYDATABUFFERPOOLSIZE Statement” on page 621
UCBPOOLSIZE	10%	5%	“UCBPOOLSIZE Statement” on page 627

Routing Statements

TCP/IP supports static and dynamic routing. Static routes to other TCP/IP hosts are defined using the GATEWAY configuration statement. Dynamic routing is provided by the routing daemon MPRoute. The following are PROFILE TCPIP statements that affect routing:

Statement	Location
ARPAGE	“ARPAGE Statement” on page 529
ASSORTEDPARMS	“ASSORTEDPARMS Statement” on page 529
GATEWAY	“GATEWAY Statement” on page 568
HOME	“HOME Statement” on page 579
LINK	“DEVICE and LINK Statements” on page 539
PATHMTUAGE	“PATHMTUAGE Statement” on page 600
PRIMARYINTERFACE	“PRIMARYINTERFACE Statement” on page 606
ROUTERADV	“ROUTERADV Statement” on page 609
ROUTERADVPREFIX	“ROUTERADVPREFIX Statement” on page 611

For details, see [Chapter 8, “Configuring the MPRoute Server,” on page 193](#).

Secure Communications Support Statements

TCP/IP can provide support for secure communications. This support is configured, in part, by the PROFILE TCPIP statements that follow:

Statement	Location
ASSORTEDPARMS	“ASSORTEDPARMS Statement” on page 529
INTERNALCLIENTPARMS	“INTERNALCLIENTPARMS Statement” on page 584
PORT	“PORT Statement” on page 603
SSLLIMITS	“SSLLIMITS Statement” on page 616
SSLSERVERID	“SSLSERVERID Statement” on page 617

In addition, a DTCPARMS file :DCSS_Parms . entry must be specified for a TCP/IP server that is to provide for secure communications support. This entry signifies to the server that secure communications support has been configured for use, and that the information provided by the SSLSERVERID and SSLLIMITS statements should be fully utilized.

More, when a :DCSS_Parms . entry is encountered by the TCP/IP server initialization program, action is taken to autolog an SSL DCSS Management Agent server (with corresponding shutdown operations performed when the TCP/IP server itself is shut down). These operations are independent of those performed through processing of any AUTOLOG statement in the TCP/IP server configuration file.

Note: Configuration of an SSL server or server pool must also be completed to provide support for secure communications. For details, see [Chapter 15, “Configuring the SSL Server,” on page 453](#).

Tracing Statements

TCP/IP provides the capability to log various events that occur in the TCPIP virtual machine. Tracing should normally be turned off, but can be requested by the TCP/IP service group. The trace output can be directed to the TCPIP virtual machine console or to a disk file. The trace-related configuration statements are:

Statement	Location
FILE	“FILE Statement” on page 565
LESSTRACE	“LESSTRACE Statement” on page 592
MORETRACE	“MORETRACE Statement” on page 595
NOSCREEN	“NOSCREEN Statement” on page 596
NOTRACE	“NOTRACE Statement” on page 597
PACKETTRACESIZE	“PACKETTRACESIZE Statement” on page 598
SCREEN	“SCREEN Statement” on page 614
TIMESTAMP	“TIMESTAMP Statement” on page 621
TRACE	“TRACE Statement” on page 623
TRACEONLY	“TRACEONLY Statement” on page 625

TCP/IP Configuration Statements

This section describes the statements you use to customize the TCP/IP stack and reflect your installation’s network configuration.

Configuration Statement Syntax

Statement syntax is the same in both the configuration file and an obey file. The following formatting restrictions apply to configuration statements:

- Statements are free format; leading blanks, comments, and end-of-record are ignored.
- A configuration statement consists of a statement name followed by a required blank and usually one or more positional arguments. Separate arguments with one or more blanks.
- A semicolon, followed by a blank, begins a comment. Comments act as blanks, separating words without affecting their meaning.
- Arguments followed by comments must have a blank before the semicolon.
- Statements can be split across multiple lines.
- Sequence numbers are not allowed.
- Lowercase letters are translated to uppercase before a statement is processed.
- Abbreviations of statement names are not allowed.
- An `ENDstatement` terminates several statements, such as `AUTOLOG` and `ASSORTEDPARMS`. If the `ENDstatement` is omitted, all subsequent tokens in the file are interpreted as parameters of that configuration statement.

Summary of TCP/IP Configuration Statements

The following table summarizes the TCP/IP configuration statements.

Table 44. Summary of TCP/IP Configuration Statements

Statement	Description	Location
ACBPOOLSIZE	Defines the initial number of activity control blocks in the free pool.	“ACBPOOLSIZE Statement” on page 527
ADDRESSTRANSLATIONPOOLSIZE	Defines the initial number of address translation control blocks in the free pool.	“ADDRESSTRANSLATIONPOOLSIZE Statement” on page 528
ARPAGE	Defines the number of minutes before an ARP table entry is deleted.	“ARPAGE Statement” on page 529
ASSORTEDPARMS	Defines miscellaneous TCP/IP parameters.	“ASSORTEDPARMS Statement” on page 529
AUTOLOG	Supplies the names of additional virtual machines to be started when TCP/IP is initialized.	“AUTOLOG Statement” on page 534
BLOCK	Specifies IP addresses from which traffic is to be blocked.	“BLOCK Statement” on page 535
CCBPOOLSIZE	Defines the initial number of client control blocks in the free pool.	“CCBPOOLSIZE Statement” on page 536
CHECKSUM	Instructs the TCPIP virtual machine to re-enable TCP checksum testing on incoming messages, if it has been disabled by the NOCHECKSUM statement.	“CHECKSUM Statement” on page 677
DATABUFFERLIMITS	Specifies the maximum number of data buffers that can be allocated for a TCP connection that uses window scaling.	“DATABUFFERLIMITS Statement” on page 537
DATABUFFERPOOLSIZE	Defines the initial number of data buffers in the free pool.	“DATABUFFERPOOLSIZE Statement” on page 537
DEVICE and LINK Channel-to-Channel Adapter	Defines an interface to a network or host.	“DEVICE and LINK statements for CTC devices” on page 539
HiperSockets Connection		“DEVICE and LINK statements for HiperSockets connections” on page 542

Table 44. Summary of TCP/IP Configuration Statements (continued)

Statement	Description	Location
Local IUCV Connections		“DEVICE and LINK statements for local IUCV connections” on page 545
Remote IUCV Connections		“DEVICE and LINK statements for remote IUCV connections” on page 548
LAN Channel Stations		“DEVICE and LINK statements for LCS devices” on page 550
OSA-Express Adapters		“DEVICE and LINK statements for OSD devices” on page 553
Virtual IP Addressing (VIPA)		“DEVICE and LINK statements for virtual IP addressing (VIPA) devices” on page 563
ENVELOPEPOOLSIZE	Defines the initial number of envelopes in the free pool.	“ENVELOPEPOOLSIZE Statement” on page 564
FILE	Specifies a file to receive trace information.	“FILE Statement” on page 565
FIXEDPAGESTORAGEPOOL	Defines the initial number of Fixed Page Storage Blocks and the maximum to be placed in the pool.	“FIXEDPAGESTORAGEPOOL” on page 566
FOREIGNIPPOOLSIZE	Defines the initial number of foreign IP address control blocks in the free pool.	“FOREIGNIPPOOLSIZE Statement” on page 568
GATEWAY	Indicates how to route datagrams to specified networks.	“GATEWAY Statement” on page 568
HOME	Defines a list of home addresses and associated link names.	“HOME Statement” on page 579
ICMPERRORLIMIT	Specifies the maximum number of ICMPv6 error messages that can be sent on a link per second.	“ICMPERRORLIMIT Statement” on page 583
INFORM	Lists users who are to be informed in case of serious run-time conditions.	“INFORM Statement” on page 584
INTERNALCLIENTPARMS	Configures the Telnet server, an internal client of TCP/IP.	“INTERNALCLIENTPARMS Statement” on page 584
IPROUTEPOOLSIZE	Defines the initial number of IP route control blocks in the free pool.	“IPROUTEPOOLSIZE Statement” on page 590
KEEPALIVEOPTIONS	Specifies the operating parameters of the TCP keep-alive mechanism.	“KEEPALIVEOPTIONS Statement” on page 590
LARGEENVELOPEPOOLSIZE	Defines the initial number of large envelopes in the free pool.	“LARGEENVELOPEPOOLSIZE Statement” on page 591
LESSTRACE	Turns off detailed tracing of specified TCP/IP processes.	“LESSTRACE Statement” on page 592
MAXRESTART	Specifies the maximum number of times the TCP/IP server will attempt to restart a user.	“MAXRESTART Statement” on page 593
MONITORRECORDS	Controls which monitor data records are generated.	“MONITORRECORDS Statement” on page 593
MORETRACE	Turns on detailed tracing of specified TCP/IP processes.	“MORETRACE Statement” on page 595
NCBPOOLSIZE	Sets the initial number of IPv6 neighbor control blocks.	“NCBPOOLSIZE Statement” on page 595
NOCHECKSUM	Instructs the TCPIP virtual machine to ignore TCP checksum errors on incoming datagrams. Checksums are generated for outgoing datagrams, regardless of whether the NOCHECKSUM statement is used.	“NOCHECKSUM Statement” on page 677
NOSCREEN	Directs trace output to the current trace file on disk.	“NOSCREEN Statement” on page 596

Table 44. Summary of TCP/IP Configuration Statements (continued)

Statement	Description	Location
NOTRACE	Turns off all tracing for specified TCP/IP processes.	“NOTRACE Statement” on page 597
OBEY	Identifies users who can use privileged TCP/IP commands and services.	“OBEY Statement” on page 597
PACKETTRACESIZE	Specifies the amount of data from each packet that is to be included in packet traces.	“PACKETTRACESIZE Statement” on page 598
PATHMTUAGE	Specifies how long path MTU discovery information is retained.	“PATHMTUAGE Statement” on page 600
PENDINGCONNECTIONLIMIT	Defines the number of half-open connections that are allowed at any given time.	“PENDINGCONNECTIONLIMIT Statement” on page 600
PERMIT	Identifies users who can use TCP/IP services.	“PERMIT Statement” on page 601
PERSISTCONNECTIONLIMIT	Defines the maximum number of connections in TCP persist state at any given time.	“PERSISTCONNECTIONLIMIT Statement” on page 602
PORT	Assigns a port to one or more servers.	“PORT Statement” on page 603
PORT Statement for Telnet	Assigns a port to the internal Telnet server.	“PORT Statement” on page 603
PRIMARYINTERFACE	Specifies which link is the primary interface.	“PRIMARYINTERFACE Statement” on page 606
RCBPOOLSIZE	Defines the initial number of Raw IP control blocks in the free pool.	“RCBPOOLSIZE Statement” on page 608
RESTRICT	Lists users who are prohibited from using TCP/IP.	“RESTRICT Statement” on page 608
ROUTERADV	Specifies whether router advertisements should be sent on a link and, if so, the minimum and maximum time between sending unsolicited router advertisements as well as other information included in router advertisements on the link.	“ROUTERADV Statement” on page 609
ROUTERADVPREFIX	Specifies prefixes that will be included in router advertisements on specific links.	“ROUTERADVPREFIX Statement” on page 611
SCBPOOLSIZE	Defines the initial number of socket control blocks in the free pool.	“SCBPOOLSIZE Statement” on page 613
SCREEN	Directs trace output to the console.	“SCREEN Statement” on page 614
SKCBPOOLSIZE	Defines the initial number of socket interface control blocks in the free pool.	“SKCBPOOLSIZE Statement” on page 614
SMALLDATABUFFERPOOLSIZE	Defines the initial number of small data buffers in the free pool.	“SMALLDATABUFFERPOOLSIZE Statement” on page 615
SOMAXCONN	Specifies the maximum length of the connection request queue created by the socket call listen().	“SOMAXCONN Statement” on page 616
SSLLIMITS	Specifies the total number of secure connections allowed and the connection limit for each SSL server.	“SSLLIMITS Statement” on page 616
SSLSERVERID	Forces the SSL server to be autologged before any other servers.	“SSLSERVERID Statement” on page 617
START	Starts the specified device.	“START Statement” on page 618
STOP	Stops the specified device.	“STOP Statement” on page 618

Table 44. Summary of TCP/IP Configuration Statements (continued)

Statement	Description	Location
SYSCONTACT	Specifies the value of the MIB-II variable sysContact, which contains information about the TCP/IP administrator for this host.	“SYSCONTACT Statement” on page 619
SYSLOCATION	Specifies the value of the MIB-II variable sysLocation, which contains information about the physical location of this host.	“SYSLOCATION Statement” on page 619
TCBPOOLSIZE	Defines the initial number of TCP control blocks in the free pool.	“TCBPOOLSIZE Statement” on page 620
TIMESTAMP	Determines how often time stamps are displayed with messages.	“TIMESTAMP Statement” on page 621
TINYDATABUFFERPOOLSIZE	Defines the initial number of tiny data buffers in the free pool.	“TINYDATABUFFERPOOLSIZE Statement” on page 621
TN3270E	Defines client IP addresses and logical unit names that may establish printer sessions.	“TN3270E Statement” on page 622
TRACE	Identifies internal TCP/IP processes for run-time tracing.	“TRACE Statement” on page 623
TRACEONLY	Restricts TCP/IP tracing to certain users, devices, or IP addresses.	“TRACEONLY Statement” on page 625
TRANSLATE	Indicates the relationship between an IP address and a network address.	“TRANSLATE Statement” on page 626
UCBPOOLSIZE	Defines the initial number of UDP control blocks in the free pool.	“UCBPOOLSIZE Statement” on page 627
VSWITCH CONTROLLER	Specifies whether a stack is available to control a CP-defined virtual switch's connection to a real LAN segment through an OSA-Express adapter. When a virtual switch is defined, CP uses a TCP/IP for z/VM stack to control its interface to the network through an OSA-Express adapter. The VSWITCH CONTROLLER statement allows the TCP/IP for z/VM stack to define where three or more control devices are attached.	“VSWITCH CONTROLLER Statement” on page 628

ACBPOOLSIZE Statement

Use the ACBPOOLSIZE statement to set the initial number of activity control blocks (ACBs). ACBs are used to schedule processes within the TCPIP virtual machine.



Operands

number

The initial number of ACBs in the free pool. The default is 1000. The minimum number is 100.

Examples

The following example shows an ACBPOOLSIZE statement that defines the number of ACBs to be the default of 1000.

```
ACBpoolSize 1000
```

Usage Notes

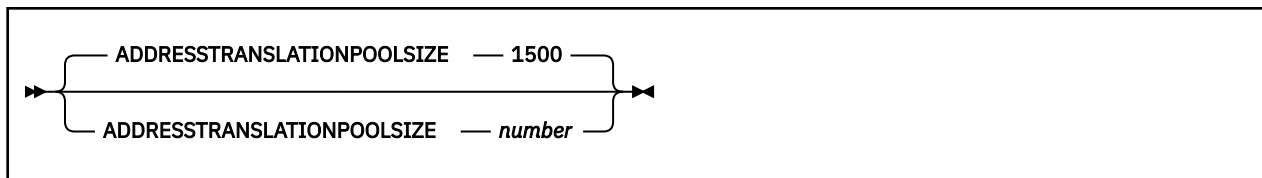
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- As long as you do not specify NOACBCUSHION on the ASSORTEDPARMS statement, the system will attempt to dynamically allocate 10% more ACBs any time the ACB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many ACBs your system is using. To avoid dynamic allocation of ACBs during operation, use the ACBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- “ASSORTEDPARMS Statement” on page 529
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

ADDRESSTRANSLATIONPOOLSIZE Statement

Use the ADDRESSTRANSLATIONPOOLSIZE statement to set the initial number of address translation control blocks. Address translation control blocks are used to hold information about the relationship between IP addresses and network addresses. Each one requires 153 bytes.



Operands

number

The initial number of address translation control blocks in the free pool.

Examples

The following example shows an ADDRESSTRANSLATIONPOOLSIZE statement that defines the number of address translation control blocks to be the default of 1500.

```
AddressTranslationPoolSize 1500
```

Usage Notes

- Each entry in the ARP table, whether entered using the TRANSLATE statement, created dynamically via ARP, or added by a device driver as a home address translation entry, requires one address translation control block.
- The system will attempt to dynamically allocate 10% more address translation control blocks any time the address translation control block free pool becomes empty. You can use the NETSTAT POOLSIZE command to monitor how many address translation control blocks your system is using. To avoid dynamic allocation of address translation control blocks during operation, use the ADDRESSTRANSLATIONPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- “TRANSLATE Statement” on page 626
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

ARPAGE Statement

Use the ARPAGE statement to determine how long an ARP table entry is retained after it is created or revalidated. By default, TCP/IP deletes ARP table entries 5 minutes after they are created or revalidated. An ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address.



Operands

minutes

The number of minutes between creation or revalidation of an ARP table entry and its deletion. The default is 5.

Examples

The following example shows an ARPAGE statement that clears the ARP tables every 5 minutes.

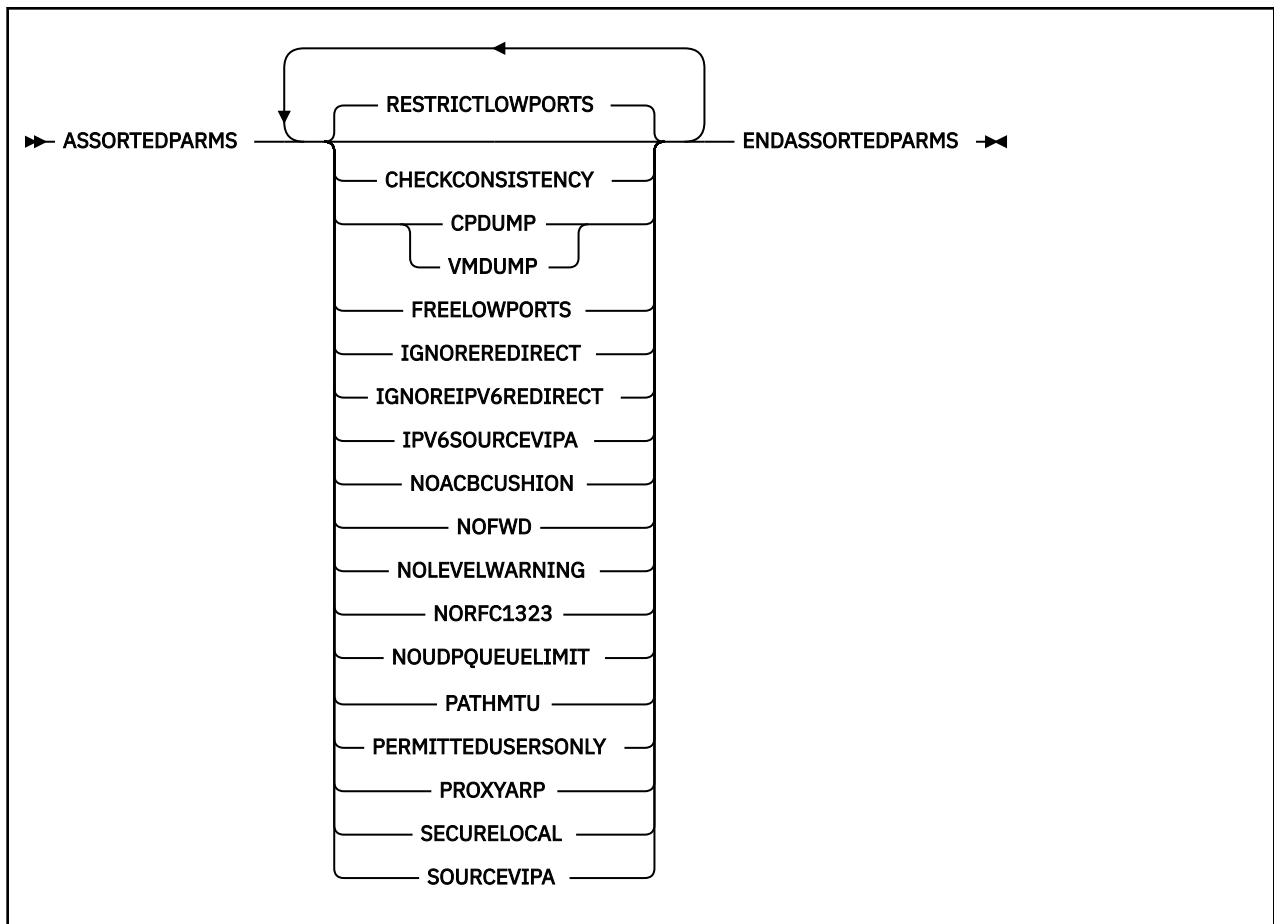
```
arpAge 5
```

Usage Notes

This number is an integer from 1 to 99,999,999.

ASSORTEDPARMS Statement

Use the ASSORTEDPARMS statement to pass initialization parameters to TCP/IP. Any misspelled or otherwise undefined parameters are flagged with a message identifying the incorrect line. A maximum of 300 characters can be used to specify the ASSORTEDPARMS.



Operands

CHECKCONSISTENCY

Consistency checking performs periodic internal checks to ensure that data structures within the TCP/IP virtual machine are intact. Making these checks necessarily involves referencing a potentially large number of pages and can cause undesirable intermittent bursts of paging activity. Specify this parameter if there is reason to believe that internal inconsistency is a source of other problems, in order to attempt to detect these earlier. The consistency check setting is confirmed with one of the messages:

```
Internal consistency checking is enabled
Internal consistency checking is disabled
```

CPDUMP

Causes any dump created as a result of a program check to be generated using the CP DUMP command.

The CPDUMP parameter is confirmed by the message:

```
TCP/IP will take dump in case of program check
```

VMDUMP

Causes any dump created as a result of a program check to be generated using the CP VMDUMP command.

The VMDUMP parameter is confirmed by the message:

```
TCP/IP will take VMDUMP in case of program check
```

If neither VMDUMP or CPDUMP is specified, no dump will be generated. This situation is confirmed with the message:

```
TCP/IP will not dump if a program check occurs
```

FREELOWPORTS

Allows the use of well-known low ports (1 through 1023). In this case, the restriction of well-known ports is controlled by the PORT statement.

If both FREELOWPORTS and RESTRICTLOWPORTS are specified, then ports 1 through 1023 are reserved.

The state of the FREELOWPORTS parameter is confirmed by one of the following messages:

```
Access to ports 1-1023 is not restricted
Only users in the obey list and users who have a port explicitly reserved may use ports
1-1023
```

IGNOREREDIRECT

(IPv4 only) Causes TCP/IP to ignore ICMP redirect packets. The IGNOREREDIRECT setting is confirmed by one of the following messages:

```
Will ignore ICMP redirects
Will honor ICMP redirects
```

If you are using MPRoute for IPv4, ICMP redirects will be ignored even when IGNOREREDIRECT is not specified.

IGNOREIPV6REDIRECT

(IPv6 only) Causes TCP/IP to ignore ICMPv6 redirect packets. The IGNOREIPV6REDIRECT setting is confirmed by one of the following messages:

```
Will ignore ICMPv6 redirects
Will honor ICMPv6 redirects
```

If you are using MPRoute for IPv6, ICMPv6 redirects are ignored even when you do not specify IGNOREIPV6REDIRECT.

IPV6SOURCEVIPA

(IPv6 only) Requests that TCP/IP to use the virtual IP address assigned to the SOURCEVIPINTERFACE of IPv6 devices as the source IP address for outbound datagrams. This parameter has no effect on interfaces which do not specify a SOURCEVIPINTERFACE.

NOACBCUSHION

Prevents dynamic expansion of the ACB pool, so that TCP/IP terminates if the pool is exhausted. Normally, when the number of free ACBs drops below 5%, the system attempts to allocate more of them. Do not specify this parameter in ordinary circumstances.

NOFWD

(IPv4 and IPv6) Stops the transfer of data between networks by disabling IPv4 and IPv6 datagram forwarding. This statement can be used for security or to ensure correct use of limited resources.

If NOFWD is not specified, IPv4 and IPv6 packets will be forwarded when this host is a gateway.

If you are using MPRoute, it is recommended that you not use this option.

The IP forwarding setting is confirmed by one of the messages:

```
IP forwarding is enabled
IP forwarding is disabled
```

NOLEVELWARNING

Suppresses messages DTCREQ076I and DTCREQ077I, which identify users running TCP/IP client programs that are not at the same level as the TCP/IP stack. If the volume of messages is too heavy, enable this option until you can upgrade the incompatible programs to the correct level.

NORFC1323

Prevents the initiation of window scaling and associated features for high-performance TCP connections, as specified by RFC 1323. If NORFC1323 is not specified, TCP/IP tries to enable window scaling and related TCP performance options for connections it initiates. Requests from other hosts to enable these facilities are always accepted.

The state of RFC 1323 support is reported by one of the messages:

```
Support for RFC 1323 is enabled
Support for RFC 1323 is disabled
```

NOUDPQUEUELIMIT

Causes TCP/IP to relax the limit of 21 incoming datagrams queued on a UDP port. If you specify NOUDPQUEUELIMIT when you are running untested applications on your system, a malfunctioning application can tie up the available envelopes.

The state of the NOUDPQUEUELIMIT parameter is confirmed by one of these messages:

```
Limit on incoming UDP datagram queue size enabled
Limit on incoming UDP datagram queue size disabled
```

Note: This parameter will be deprecated in the next release. Use the UDPQUEUELIMIT statement to set a hard limit. For more information, see [“UDPQUEUELIMIT Statement” on page 627](#).

PATHMTU

(IPv4 only) Enables path MTU discovery as the default value for links which have not explicitly specified PATHMTU or NOPATHMTU on the LINK statement. If omitted, links not explicitly configured for path MTU discovery will have path MTU discovery disabled.

PERMITTEDUSERONLY

Restricts the use of TCP/IP services to only those users who are explicitly identified in the initial configuration file or any subsequent obey file using the PERMIT statement.

Without this parameter, any user not specified on the RESTRICT statement can use TCP/IP services.

Use of the PERMITTEDUSERONLY operand is confirmed by the following message:

```
Only users mentioned in PROFILE TCPIP may use TCP/IP services
```

When the PERMITTEDUSERONLY operand is not used and there are users specified in the RESTRICT list, the following message is issued:

```
All users except those in the RESTRICT list may use TCP/IP services
```

When the PERMITTEDUSERONLY operand is not used and there are no users specified in the RESTRICT list, then the following message is issued:

```
Access to TCP/IP services is not restricted
```

PROXYARP

(IPv4 only) Causes the z/VM system where TCP/IP is running to act as a proxy for guest hosts that are defined with static host routes. ARP requests for the MAC address associated with the IP address of such hosts are answered by TCP/IP on behalf of the guest. For QDIO and Hipersockets devices, the IP addresses of such hosts are added to the list of those whose traffic is handled by TCP/IP for z/VM. Whether through proxy ARP or IP address advertisement, traffic for these hosts is routed to z/VM for forwarding to the ultimate destination.

You must specify PROXYARP when the IP address for a host is in the same subnet as other IP addresses in use on LANs to which the z/VM host (that is providing guest support) has a connection.

RESTRICTLOWPORTS

Restricts the use of well-known ports (1 through 1023) to users who are specified on the OBEY statement and to users who have a port explicitly reserved for them on a PORT statement. RESTRICTLOWPORTS is the default.

If FREELOWPORTS is specified, then the restriction of well-known ports is controlled exclusively by the PORT statement.

The state of the RESTRICTLOWPORTS parameter is confirmed by one of the following messages:

```
Only users in the obey list and users who have a port explicitly reserved may use ports
1-1023
Access to ports 1-1023 is not restricted
```

SECURELOCAL

Causes TCP/IP to check if local connections are destined for a secure port or user ID. If so, the connection is routed to the SSL server. If this operand is not specified, local connections are bypassed for static SSL processing. This setting has no effect on connections secured dynamically with TLS.

SOURCEVIPA

(IPv4 only) Requests TCP/IP to use the closest virtual IP address to the actual destination address in the HOME list as the source IP address for outbound datagrams. This parameter has no effect on RIP servers for routing protocol packets.

Examples

The following example shows two parameters on the ASSORTEDPARMS statement.

```
AssortedParms
NOFWD
RestrictLowPorts
EndAssortedParms
```

Usage Notes

- The following table shows the ASSORTEDPARMS options that are related to specific IP protocols: IPv4, IPv6, or both. Options not specific to IP protocols are not shown.

Table 45. Relationship of ASSORTEDPARMS options to IP protocols

Option	IPv4	IPv6
IGNOREREDIRECT	yes	no
IGNOREIPV6REDIRECT	no	yes
NOFWD	yes	yes
PATHMTU	yes	no
PROXYARP	yes	no
SOURCEVIPA	yes	no

- The ENDASSORTEDPARMS statement is the delimiter for the ASSORTEDPARMS statement. If ENDASSORTEDPARMS is omitted, subsequent statements will be processed as invalid ASSORTEDPARMS with error messages generated for each.
- When using the OBEYFILE command to modify the ASSORTEDPARMS statement, keep in mind these rules:
 - The values of all parameters are changed.
 - An ASSORTEDPARMS parameter that is not specified assumes its default value or setting.

More Information

- [“PORT Statement” on page 603](#)
- [“INFORM Statement” on page 584](#)
- [“OBEY Statement” on page 597](#)

- “RESTRICT Statement” on page 608
- “PERMIT Statement” on page 601
- “OBEYFILE Command” on page 644

AUTOLOG Statement

The AUTOLOG statement identifies other virtual machines to be started by the TCP/IP virtual machine when it begins execution.

The first AUTOLOG statement of a configuration file replaces the existing AUTOLOG list. Subsequent AUTOLOG statements in the same file add to the list.



Operands

user_id

The name of a virtual machine that the TCP/IP virtual machine should autolog. If a user ID is not valid, TCP/IP will display an error message.

0

A constant. For compatibility with prior releases of TCP/IP for VM, any arbitrary string is accepted.

Examples

The following example shows how to include two servers in the AUTOLOG statement.

```

Autolog
  FTPSERVE 0      ; FTP Server
  SMTP      0      ; SMTP Server
EndAutolog
  
```

Usage Notes

- The ENDAUTOLOG statement specifies the end of the AUTOLOG list.
- A virtual machine in the AUTOLOG list is terminated and restarted by TCP/IP whenever the stack is restarted. TCP/IP uses the CP FORCE command to terminate a virtual machine and the CP XAUTOLOG command to start a virtual machine.
- For servers connected to the stack via VMCF, TCP/IP frequently checks to ensure that each virtual machine in the AUTOLOG statement is logged on.
- The number of times the TCP/IP stack will restart a particular server is governed by the MAXRESTART statement.
- If the AUTOLOG list is empty or if the AUTOLOG statement is omitted, TCP/IP does not attempt to start any of the higher-level servers, such as FTP, MPRoute, or REXECD. Each server must be started manually.
- Once a virtual machine in the AUTOLOG makes a connection to the TCP/IP stack, it is expected to maintain its connection. If all of the connections from a virtual machine are terminated, the stack assumes that a problem has occurred and attempts to reestablish the connection (by terminating and then restarting the virtual machine in question).
- If a virtual machine in the AUTOLOG list also has a PORT statement reserving a TCP or UDP port, but does not have a listening connection or is not accepting packets on that port, TCP/IP attempts to restart that virtual machine.

If the PORT statement for a virtual machine specifies NOAUTOLOG, this restart will not be attempted. Please note that NOAUTOLOG will not prevent a restart of the virtual machine stemming from circumstances unrelated to the termination of the listening connection.

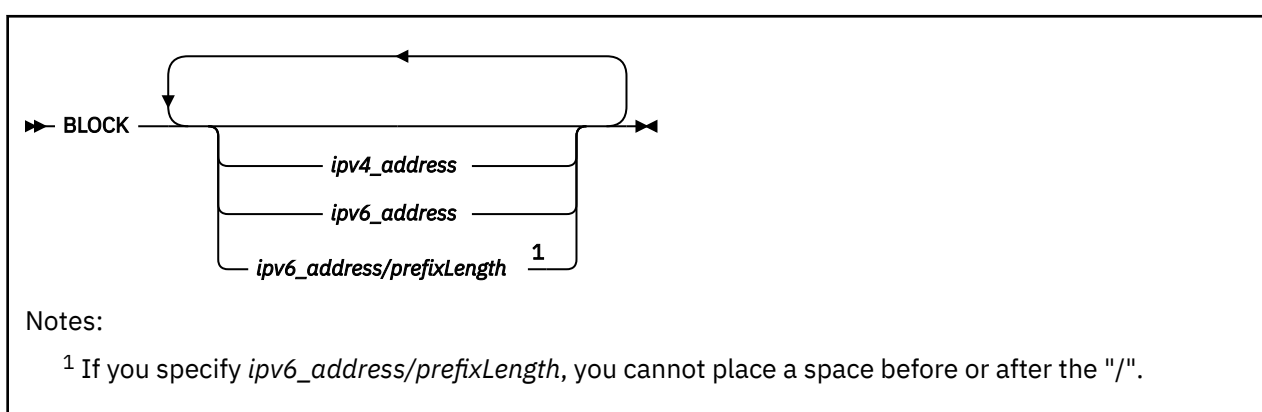
More Information

- “MAXRESTART Statement” on page 593
- “PORT Statement” on page 603

BLOCK Statement

The BLOCK statement specifies IP addresses from which traffic is to be blocked. Any packets from blocked IP addresses are filtered out of the incoming data stream and are ignored.

The first BLOCK statement of a configuration file replaces the existing packet filters definitions. If a syntax error is found in a BLOCK statement, the remainder of the statement is ignored. Subsequent BLOCK statements in the same profile or OBEYFILE add filters to the existing list.



Operands

ipv4_address

The IPv4 address in dotted-decimal form. To block traffic from multiple IPv4 addresses, specify the last token of IPv4 address as an asterisk (*).

ipv6_address

The IPv6 address in either the preferred or compressed form.

Example:

Preferred form:

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

Compressed form:

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

ipv6_address/prefixlength

The IPv6 address in either the preferred or compressed form followed by a "/" and the length of the prefix.

Example:

Preferred form:

```
1080:0:0:0:8:800:200C:417A/80
```

Compressed form:

CCBPOOLSIZE

```
1080::8:800:200C:417A/80
```

If the IPv6 address of an incoming packet matches the designated set of IPv6 addresses, the packet is filtered out. Using the *ipv6_address/prefixlength* of the previous example, packets from IP addresses 1080:0:0:0:8:0:0:0 through 1080:0:0:0:8:FFFF:FFFF:FFFF would be filtered out.

Examples

1. The following example shows how to block packets from IP address 9.130.58.78.

```
BLOCK 9.130.58.78
```

2. The following example shows how to block packets 9.130.58.0 through 9.130.58.255.

```
BLOCK 9.130.58.*
```

3. The following example shows how to block packets from network 9.

```
BLOCK 9.*
```

4. The following example shows how to block IPv6 packets from a host with an IP address of 5014:c2c1::200:0:18:c00 and from the subnet 5014:c2c1::/32.

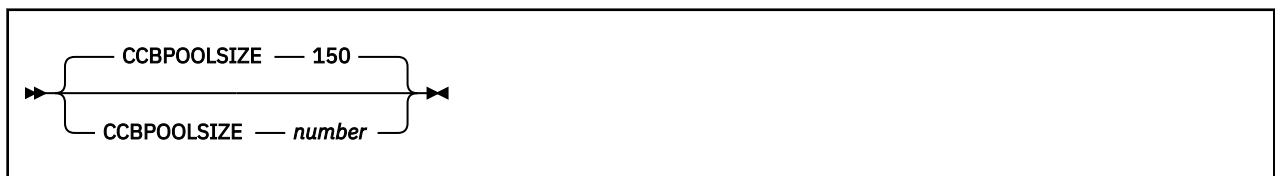
```
BLOCK  
5014:c2c1::200:0:18:c00  
5014:c2c1::/32
```

Usage Notes

- To block a range of IPv6 addresses (for example, a subnet), use the *ipv6_address/prefixlength* format. This format is equivalent to specifying the asterisk (*) when blocking a range of IPv4 addresses.
- The NETSTAT command can be used to display and change the list of blocked IP addresses and to determine how many packets have been blocked.

CCBPOOLSIZE Statement

Use the CCBPOOLSIZE statement to set the initial number of client control blocks (CCBs). A CCB is needed for each virtual machine using the TCPIP virtual machine, including servers.



Operands

number

The initial number of CCBs in the free pool. The default is 150.

Examples

The following example shows a CCBPOOLSIZE statement that defines the number of CCBs to be the default of 150.

```
CCBpoolSize 150
```

Usage Notes

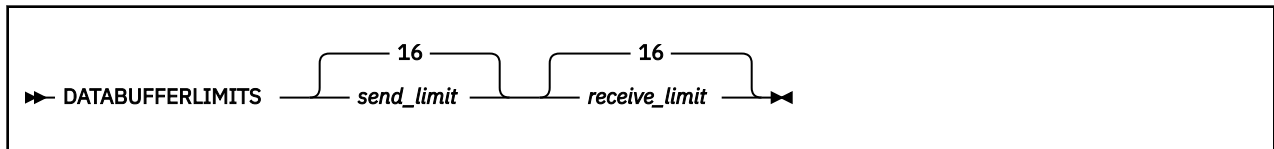
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- When running with ASSORTEDPARMS PERMITTEDUSERONLY, every user mentioned in the configuration file uses a CCB. Otherwise, only the users mentioned in an OBEY or INFORM statement use a CCB. The CCB is used even if the user is not actively using TCP/IP services.
- The system attempts to dynamically allocate 10% more CCBs any time the CCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many CCBs your system is using. To avoid dynamic allocation of CCBs during operation, use the CCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [“INFORM Statement” on page 584](#)
- [“OBEY Statement” on page 597](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

DATABUFFERLIMITS Statement

Use the DATABUFFERLIMITS statement to specify the maximum number of data buffers that can be allocated for a TCP connection that is using window scaling.



Operands

send_limit

The maximum number of buffers to hold outbound data.

receive_limit

The maximum number of buffers to hold inbound data. This limit, when multiplied by the regular data buffer size (see [“DATABUFFERPOOLSIZE Statement” on page 537](#)) determines the maximum receive window size that is advertised.

Usage Notes

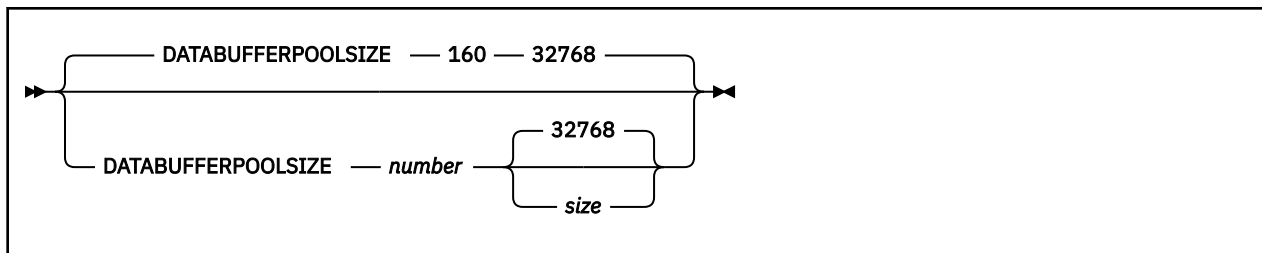
- This statement affects only connections with other hosts that also support RFC 1323.

More Information

- [“DATABUFFERPOOLSIZE Statement” on page 537](#)

DATABUFFERPOOLSIZE Statement

Use the DATABUFFERPOOLSIZE statement to set the initial number and size of regular data buffers. Regular data buffers are used by the TCP layer for connections.



Operands

number

The initial number of regular data buffers in the free pool. The default is 160.

size

The size of each regular data buffer (in bytes). The default size is 32768.

Only one of the following values or their alternates (shown in parentheses) can be specified:

8192 (8K) 49152 (48K)
 12288 (12K) 65536 (64K)
 16384 (16K) 98304 (96K)
 24576 (24K) 131072 (128K)
 28672 (28K) 196608 (192K)
 32768 (32K) 262144 (256K)

A value of 32768 or 32K optimizes data transfer for FTP clients that support window sizes greater than 8,192.

Examples

1. The following example shows a DATABUFFERPOOLSIZE statement that defines the default of 160 buffers, each 32768 bytes in length.

```
DataBufferPoolSize 160 32768
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system attempts to dynamically allocate 10% more regular data buffers any time the regular data buffer free pool level drops below 5%. You can use the NETSTAT POOLSIZE command to monitor how many regular data buffers your system is using. To avoid dynamic allocation of regular data buffers during operation, use the DATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- The TCP layer for Telnet uses regular data buffers only if there are no small data buffers available. The Telnet server also uses regular data buffers for internal processing, regardless of whether small data buffers are available.
- Increasing the size of regular data buffers usually improves FTP throughput significantly.
- For Telnet connections, the size of regular data buffers might need to be increased in order to accommodate clients that use large screen sizes.

More Information

- [“DATABUFFERLIMITS Statement” on page 537](#)
- [“SMALLDATABUFFERPOOLSIZE Statement” on page 615](#)

- “TINYDATABUFFERPOOLSIZE Statement” on page 621
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

DEVICE and LINK Statements

This section describes the following:

- Intelligent default MTU values
- DEVICE and LINK statements, which are grouped together by device type.

Intelligent Default MTU Values Based on the Device and Link Type

TCP/IP uses intelligent default MTU values whenever a LINK MTU is specified as 0. The following table shows the intelligent default MTU value for each device and link type.

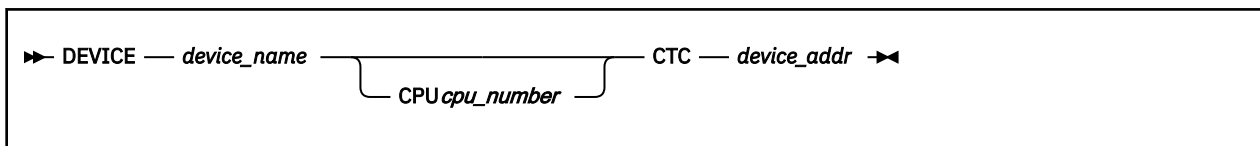
Device Type	Link Type	Intelligent Default MTU Value
CTC	CTC	min(LARGEENVELOPEPOOLSIZE, 32760)
EQDIO	EQDIOETHERNET	min(LARGEENVELOPEPOOLSIZE, 9000)
HIPERS	QDIOIP	min(LARGEENVELOPEPOOLSIZE, 4000)
IUCV	IUCV	min(LARGEENVELOPEPOOLSIZE, 32764)
LCS	ETHERNET	min(LARGEENVELOPEPOOLSIZE, 1500)
	802.3	min(LARGEENVELOPEPOOLSIZE, 1500)
	ETHEROR802.3	min(LARGEENVELOPEPOOLSIZE, 1500)
OSD	QDIOETHERNET	min(LARGEENVELOPEPOOLSIZE, 1500)
PVMIUCV	IUCV	min(LARGEENVELOPEPOOLSIZE, 32764)

Note: You can use OSD or OSH statements to configure a TCP/IP EQDIO connection.

DEVICE and LINK statements for CTC devices

DEVICE Statement — CTC Devices

Use the DEVICE statement to specify the name and device address of each channel-to-channel (CTC) device that you use.



Operands

device_name

A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CPU cpu_number

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

CTC

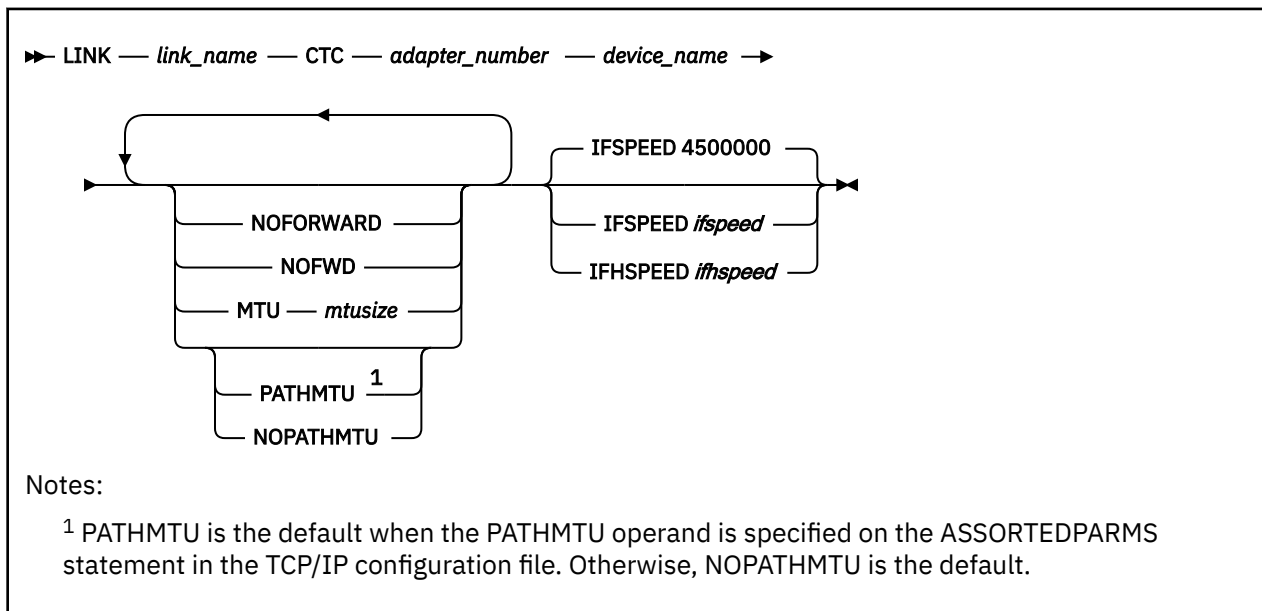
Specifies the device is a channel-to-channel device.

device_addr

The hexadecimal device address associated with the CTC adapter. TCP/IP uses *device_addr* and *device_addr*+1. If you are using HCD on OS/390® or z/OS, these devices must be defined as "CTCA" on the VM side, and "SCTC" on the OS/390 or z/OS side.

LINK Statement — CTC Devices

Use the LINK statement to define the point-to-point network interface to the remote host.



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

CTC

Specifies that the link is a channel-to-channel adapter.

adapter_number

Must be 0 or 1. Used to specify which device address is the read address and which is the write address. Use 0 to indicate that *device_addr* is read, and 1 to indicate *device_addr* is write.

device_name

The *device_name* must be the one specified in the DEVICE statement.

NOFORWARD

NOFWD

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement or 32760, whichever is lower.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtusize* to the intelligent default based on the link type as shown in [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was an intelligent default assigned by TCP/IP. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU

NOPATHMTU

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a link type is 0; the maximum value is 2147483647 bits per second. The default is 4500000 bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

IFHSPEED *ifhspeed*

An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a link type is 0. The default is 4 million bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

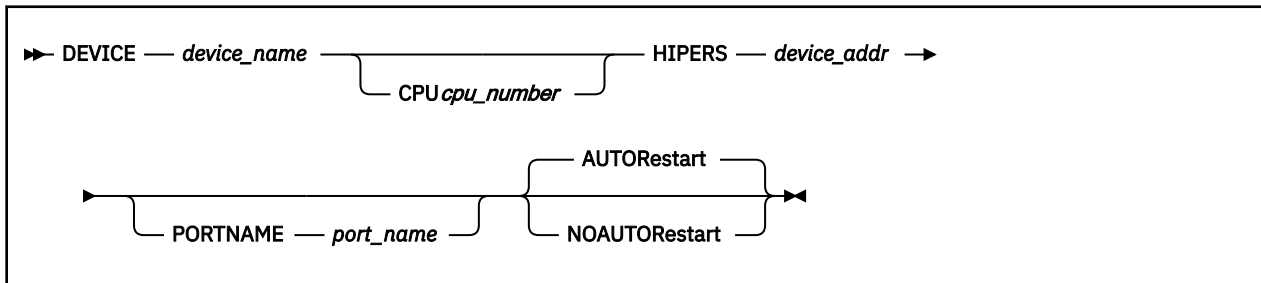
More Information

- [“Changing the TCP/IP Configuration with the OBEYFILE Command”](#) on page 644
- [“Point-to-point connections to other hosts”](#) on page 520

DEVICE and LINK statements for HiperSockets connections

DEVICE Statement — HiperSockets Connection

Use the DEVICE statement to specify the name and address of the device to be used for a HiperSockets connection.



Operands

device_name

A unique name for the device whose maximum length is 16 characters. The name is referenced in the LINK statements.

CPU *cpu_number*

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

HIPERS

Specifies that the device is a HiperSockets connection using the Queued Direct I/O Hardware Facility.

device_addr

The hexadecimal device address that specifies the first of three consecutive virtual device numbers to be grouped for the HiperSockets connection. TCP/IP uses *device_addr*, *device_addr*+1 and *device_addr*+2.

PORTNAME *port_name*

The eight character name used to identify this instance of the HiperSockets connection. If not specified, no portname will be sent to the HiperSockets connection. For details, see [“Usage Notes” on page 545](#).

AUTOREstart

Instructs the TCP/IP server to attempt to restart the device in the event of a device failure. AUTOREstart will only be attempted after successful data transfer has occurred. By default, TCP/IP restarts devices that have failed.

NOAUTOREstart

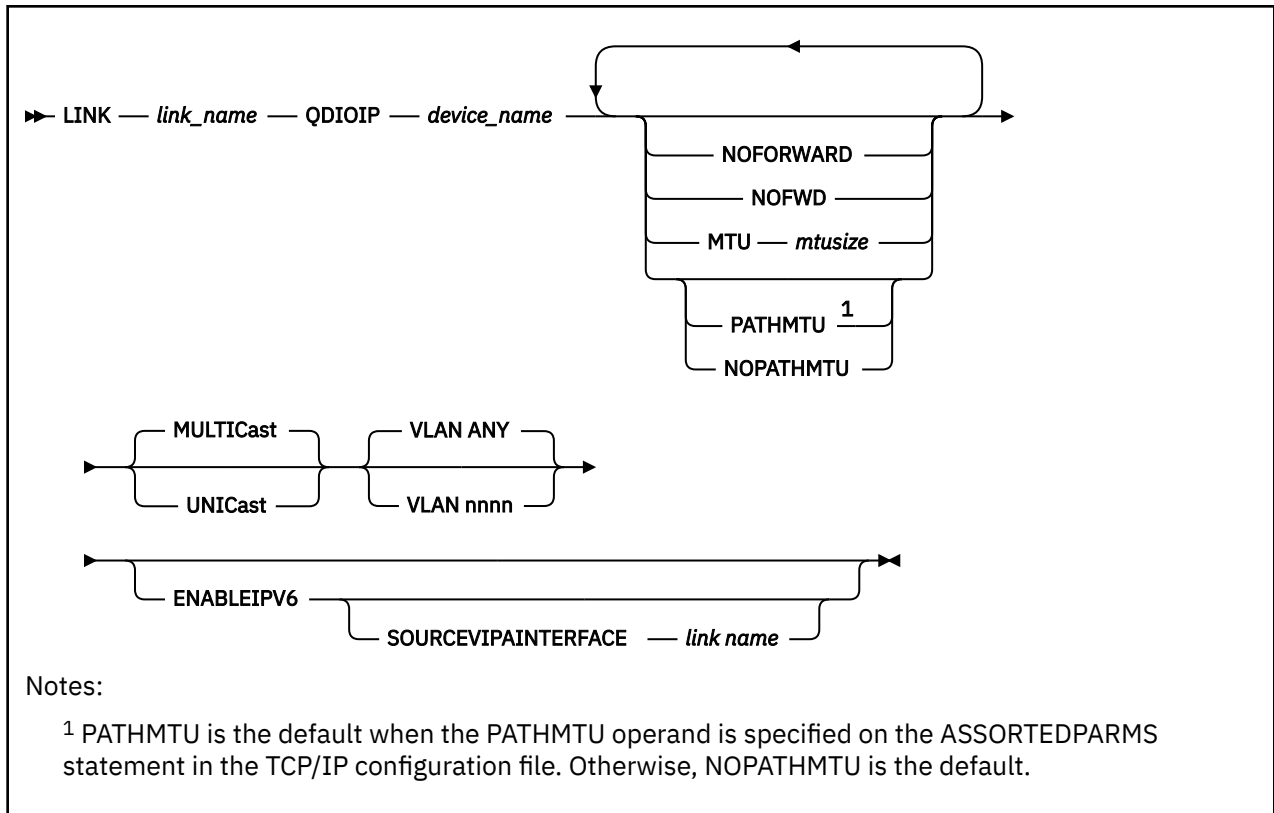
Instructs the TCP/IP server to not attempt to restart the device.

Usage Notes

As of z/VM version 5 release 4.0, the default was changed from NOAUTORESTART to AUTORESTART.

LINK Statement — HiperSockets Connection

Use the LINK statement to define the network interface for each HiperSockets device.



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

QDIOIP

Specifies that the link uses a QDIO (Queued Direct I/O) protocol for a HiperSockets connection.

device_name

The *device_name* must be the one specified in the DEVICE statement.

NOFORWARD

NOFWD

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type” on page 539](#).
- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

Note: The IPv6 intelligent default MTU value is 1500 bytes. When a LINK is enabled for IPv6 and the *mtusize* value specified is less than 1280 bytes, the IPv6 MTU value used for the link is set to 1280 bytes.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file
- IPv6 router advertisements

Note: IPv6 neighbor discovery routes learned by router advertisements use the LINK MTU *mtusize* before using the MTU option in an advertisement. If neither are available, then the value 1280 is used.

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the if MTU value in TCP/IP MONITOR records and the *mtusize* to the intelligent default based on the link type as shown in [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows 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. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS may not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that may be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU**NOPATHMTU**

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default. These operands have no effect on IPv6 routes. Path MTU discovery is always enabled for IPv6 and cannot be disabled.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that may be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

MULTIcast

Can be specified for compatibility with previous releases, but it is ignored. The capabilities of the device are automatically detected and configured.

UNICAST

Can be specified for compatibility with previous releases, but it is ignored. The capabilities of the device are automatically detected and configured.

VLAN ANY

Specifies that no identifier for a Virtual Local Area Network (VLAN) is to be associated with this interface. This is the default.

VLAN nnnn

Specifies the identifier for a Virtual Local Area Network (VLAN). *nnnn* represents a number from 1 to 4094.

When **ENABLEIPv6** is specified, **VLAN nnnn** applies to both IPv4 and IPv6 for a QDIOIP LINK. Only one VLAN ID is allowed.

ENABLEIPv6

Specifies that the link should allow IPv6 traffic.

SOURCEVIPAINTERFACE *link name*

Specifies the LINK name of the virtual interface whose address(s) are to be used as the source IP address for outgoing datagrams on this link. The interface specified must be a virtual interface which has been enabled for IPv6. If the interface specified has multiple IP addresses defined, the source address will be selected from among those addresses using the default source address selection algorithm.

Examples

In the following example, HIPR1 is a HiperSocket device. The home IP address of the z/VM host is 125.0.0.27. TCP/IP is using device addresses 1D00 through 1D02.

```
DEVICE HIPR1 HIPERS 1D00 PORTNAME REDOCT
LINK QDIO1 QDIOP HIPR1
HOME 125.0.0.27 QDIO1
```

Usage Notes

- When the device is started and a PORTNAME has been specified on the DEVICE statement, TCP/IP will set the PORTNAME as the hardware adapter name.
- The device address specified on the HIPERSockets device statement is the first of three consecutive device addresses to be grouped for the HiperSockets connection.
- If you specify ENABLEIPv6 and the hardware does not support IPv6 protocol, then it operates according to IPv4 protocol, ignoring any IPv6-related options for this link.
- When defining a device for a z/VM virtual LAN, you must establish connectivity between the device and the target LAN. This can be accomplished with the CP command CP COUPLE which will tie the device NIC to the target LAN. For more information on this command see the [z/VM: CP Commands and Utilities Reference](#).
- VLAN configuration information can be found in [Planning for Guest LANs and Virtual Switches in z/VM: Connectivity](#).

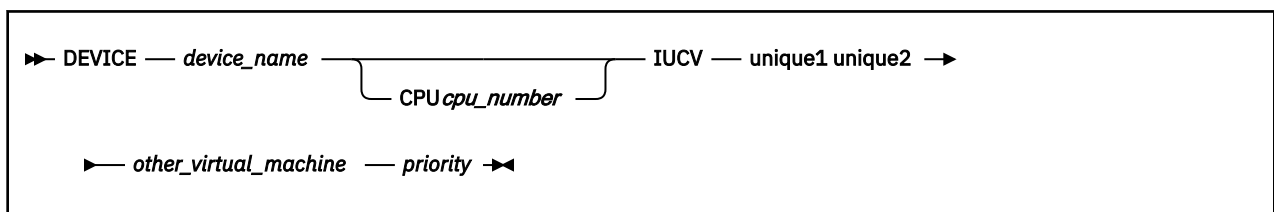
DEVICE and LINK statements for local IUCV connections

DEVICE Statement — Local IUCV Connections

Use the DEVICE statement to define an IUCV connection to another virtual machine on the same VM system that is running a TCP/IP for VM stack.

Use the LINK statement to define the point-to-point network interface to another TCP/IP stack. You can also create multiple local IUCV connections to the same TCP/IP stack. See “Usage Notes” on page 548.

The other TCP/IP stack must have a corresponding pair of DEVICE and LINK statements.



Operands

device_name

A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CPU *cpu_number*

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

IUCV

Specifies that the device is using an IUCV connection.

unique1 unique2

Specifies two 1- to 8-character identifiers. The identifiers (for example, 1 1), when concatenated with the user ID of the *other_virtual_machine*, must produce a unique value for this TCP/IP stack. You must specify the same two values for the corresponding device on the *other_virtual_machine*'s TCP/IP stack.

other_virtual_machine

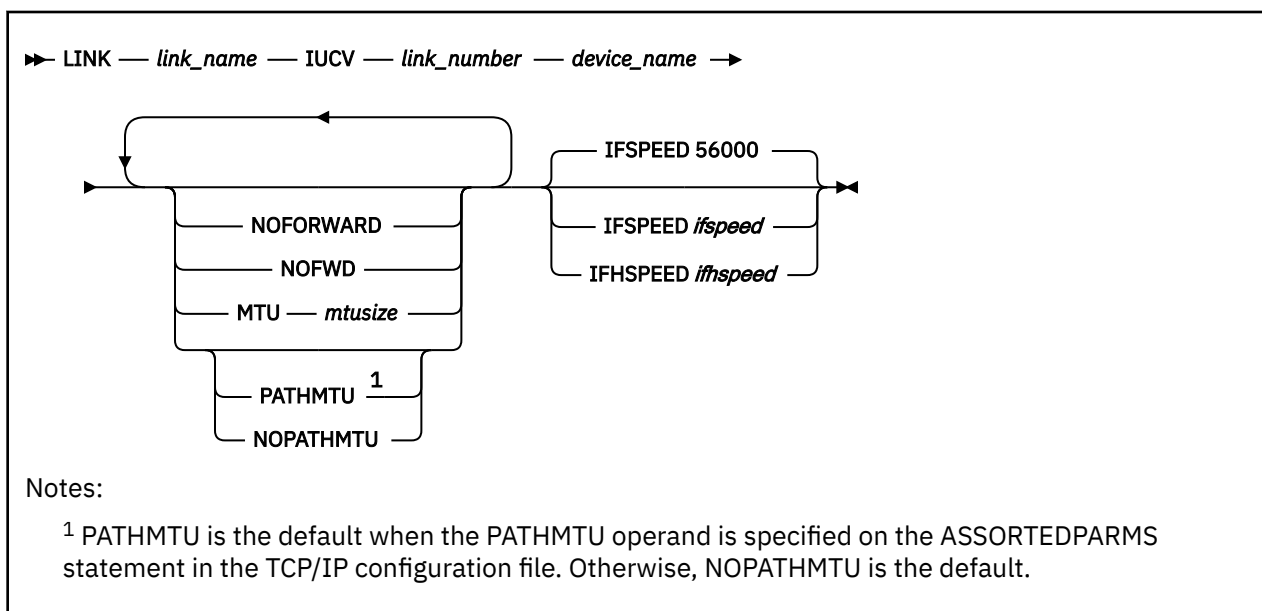
The name of another virtual machine running TCP/IP for VM to which you want to establish a connection.

priority

The order of priority between the two connected virtual machines. Use A on one virtual machine and B on the other.

LINK Statement — Local IUCV Connections

Use the LINK statement to define the point-to-point network interface to another TCP/IP stack.



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

IUCV

Specifies the device is using an IUCV connection.

link_number

Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statements for other device types.

device_name

The *device_name* must be the one specified in the DEVICE statement.

NOFORWARD**NOFWD**

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU mtusize

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type” on page 539](#).
- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtusize* to the intelligent default based on the link type as shown in [“Intelligent Default MTU Values Based on the Device and Link Type” on page 539](#). However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was an intelligent default assigned by TCP/IP. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type” on page 539](#).

PATHMTU**NOPATHMTU**

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a link type is 0; the maximum value is 2147483647 bits per second. The default is 56000 bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

IFHSPEED *ifhspeed*

An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a link type is 0. The default is 0. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

Usage Notes

- You can create multiple local IUCV connections to the same TCP/IP stack by specifying the DEVICE statement multiple times, each time with different values for *unique1 unique2*. Each DEVICE statement must have its corresponding LINK statement and the same two values must be specified for the associated device on the other stack.

More Information

- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)
- [“Point-to-point connections to other hosts” on page 520](#)

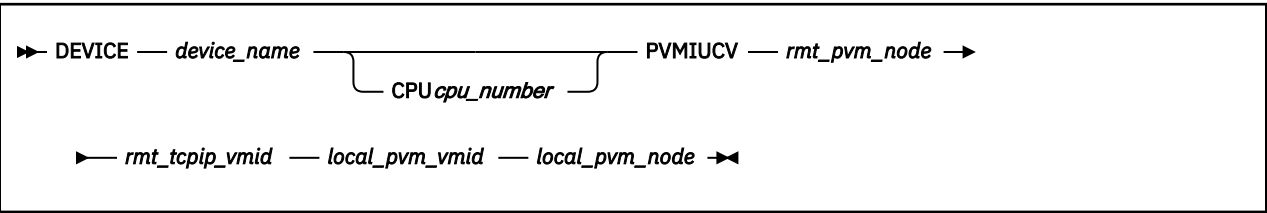
DEVICE and LINK statements for remote IUCV connections

DEVICE Statement — Remote IUCV Connections

Use the DEVICE statement to define an IUCV connection to a virtual machine on a remote VM system that is running the TCP/IP for VM stack.

The connection is established using the Personal Computer Communications Facility (PCCF) of VM/Pass-Through Facility 1.4 or later. This facility is usually referred to as "PVM IUCV".

The remote TCP/IP stack must have a corresponding pair of DEVICE and LINK statements.



Operands

device_name

A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CPU *cpu_number*

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

PVMIUCV

Specifies that the device connection is to another VM system using PVM IUCV.

rmt_pvm_node

The PVM network node name of the remote node.

rmt_tcpip_vmid

The name of the virtual machine of the TCPIP virtual machine on the remote node.

local_pvm_vmid

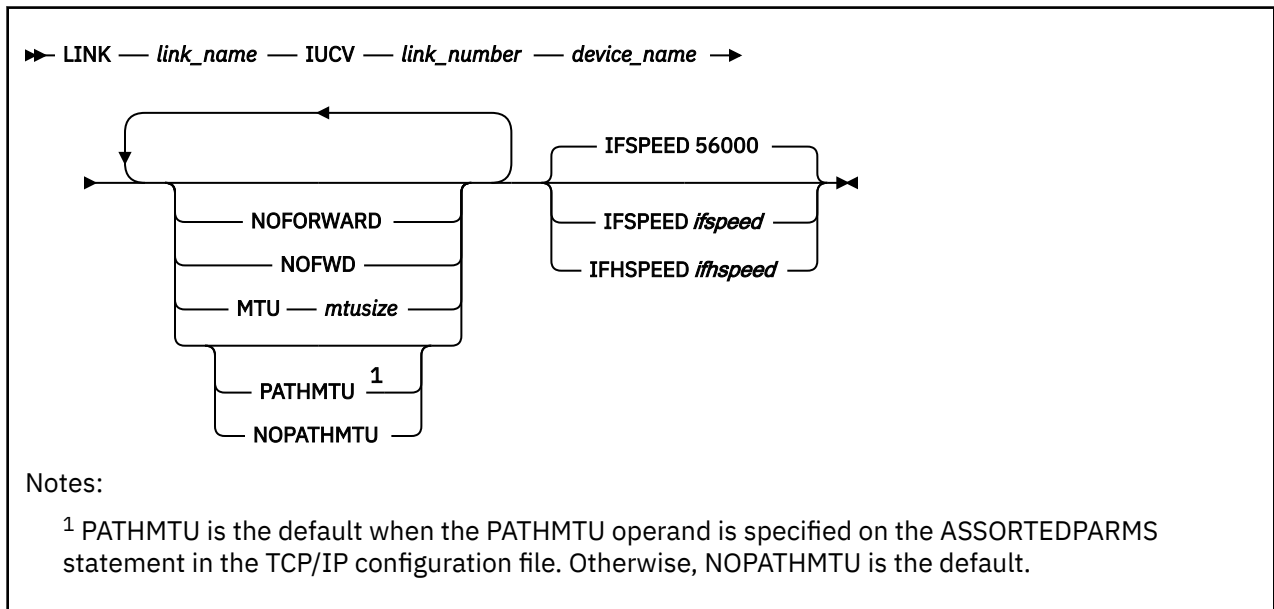
The name of the virtual machine of the PVM server on the local node.

local_pvm_node

The PVM network node name of the local node.

LINK Statement — Remote IUCV Connections

Use the LINK statement to define the point-to-point network interface to the remote TCP/IP stack.



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

IUCV

Specifies the device is using an IUCV connection.

link_number

Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name

The *device_name* must be the one specified in the DEVICE statement.

NOFORWARD

NOFWD

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtusize* to the intelligent default based on the link type as shown in [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was an intelligent default assigned by TCP/IP. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU

NOPATHMTU

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a link type is 0; the maximum value is 2147483647 bits per second. The default is 56000 bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

IFHSPEED *ifhspeed*

An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a link type is 0. The default is 0. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

More Information

- [“Changing the TCP/IP Configuration with the OBEYFILE Command”](#) on page 644
- [“Point-to-point connections to other hosts”](#) on page 520

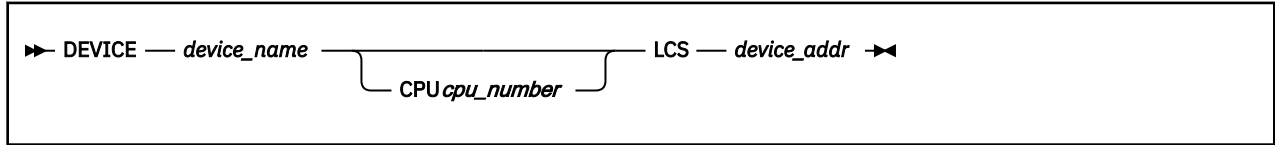
DEVICE and LINK statements for LCS devices

DEVICE Statement — LCS Devices

Use the DEVICE statement to specify the name and device address of each Open Systems Adapter in LAN Emulation mode.

Use the LINK statement to define the network interface for each Ethernet network to which the LCS is attached.

There can be as many LINK statements for each LCS as there are LAN interfaces in the LCS.



Operands

device_name

A unique name for the device. The maximum length is 16 characters. The same name is specified on the LINK statements.

CPU *cpu_number*

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

LCS

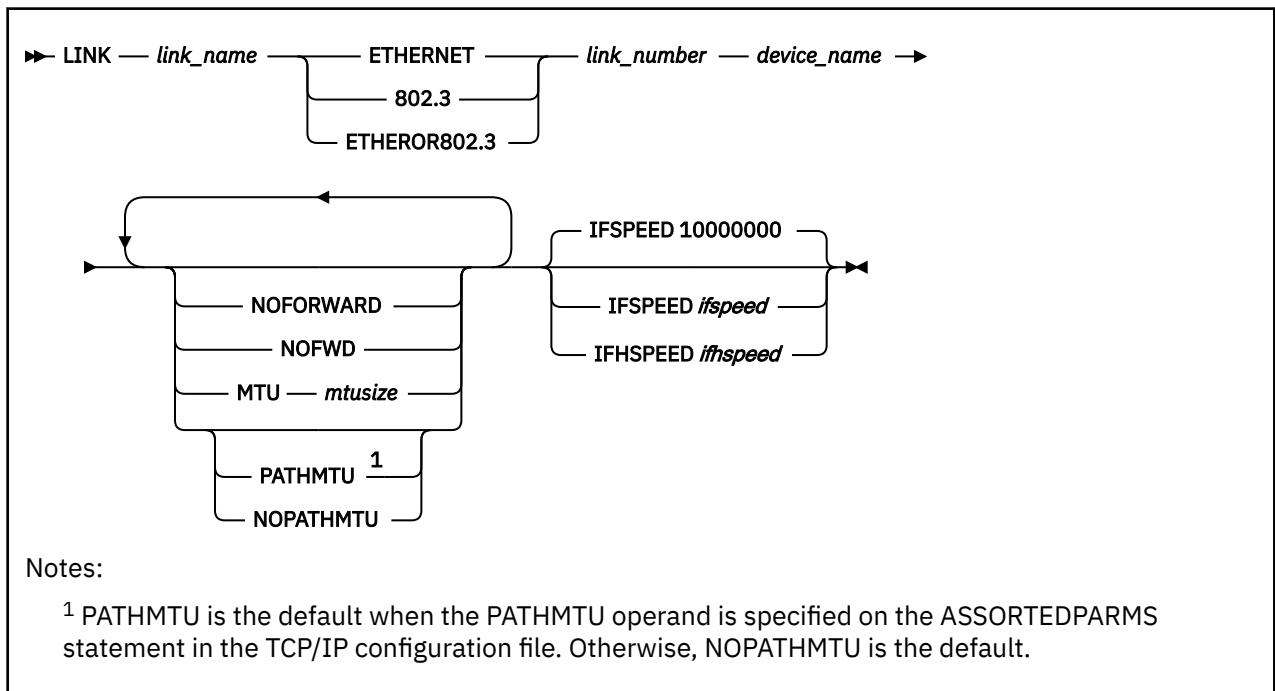
Specifies the device is a LAN Channel Station.

device_addr

The hexadecimal address of the LCS. TCP/IP uses *device_addr* and *device_addr*+1.

LINK Statement for Ethernet Network LCS

This LINK statement is used to define the Ethernet adapter on an LCS previously defined by an LCS DEVICE statement.



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

ETHERNET

Standard Ethernet protocol only.

802.3

IEEE 802.3 protocol only.

ETHEROR802.3

Both standard Ethernet and IEEE 802.3 protocols.

link_number

The relative Ethernet adapter number within the LCS. This number is assigned by the Interconnect Controller Program (ICP).

device_name

The *device_name* must be the same name as specified in the DEVICE statement.

NOFORWARD

NOFWD

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.
- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtusize* to the intelligent default based on the link type as shown in [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was an intelligent default assigned by TCP/IP. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLNKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU

NOPATHMTU

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a link type is 0; the maximum value is 2147483647 bits per second. The default is 10000000 bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

IFHSPEED *ifhspeed*

An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a link type is 0. The default is 10 million bits per second. This value is accessible to SNMP for management queries, but has no effect on the operation of the device.

Examples

1. In the following example, LCS2 is an OSA-Express2 adapter configured in LCS mode with one Ethernet adapter.

```
DEVICE LCS2    LCS          BA0
LINK ETH1  ETHERNET 0 LCS2
```

More Information

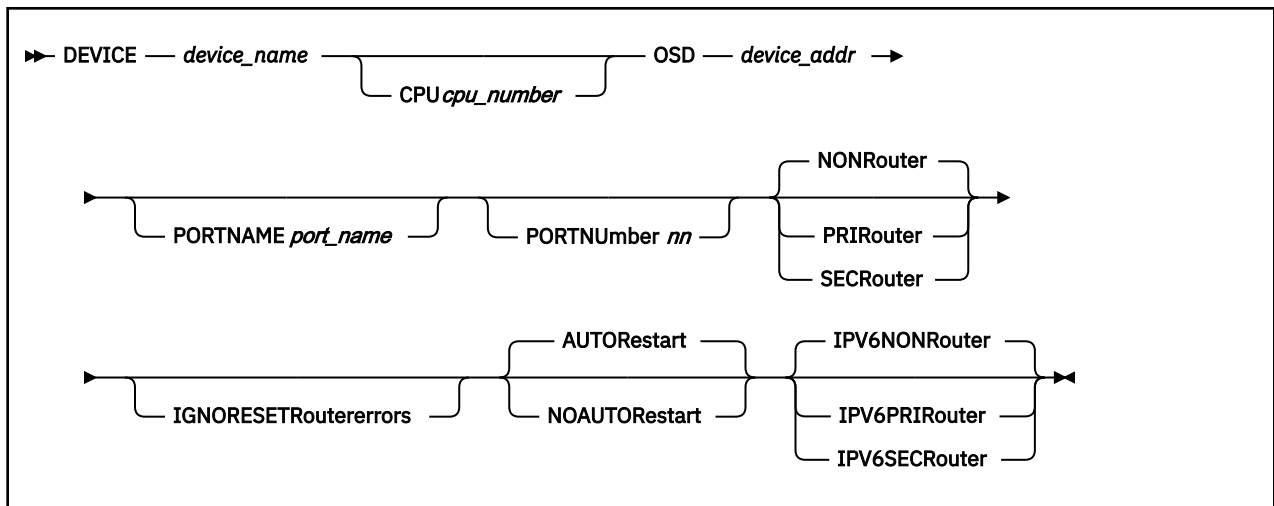
- [“GATEWAY Statement”](#) on page 568
- [“HOME Statement”](#) on page 579
- [“START Statement”](#) on page 618
- [“Changing the TCP/IP Configuration with the OBEYFILE Command”](#) on page 644

DEVICE and LINK statements for OSD devices

Note: You can use OSD or OSH statements to configure a TCP/IP EQDIO connection.

DEVICE Statement — OSD Devices

Use the DEVICE statement to specify the name and address of the device that will use the Queued Direct I/O (QDIO) Hardware Facility.



Operands

device_name

A unique name for the device whose maximum length is 16 characters. This name is referenced in the LINK statement.

CPU cpu_number

An integer between 0 and 6 which designates the virtual processor that is used to run the device driver for the associated device.

OSD

Specifies that the device is an OSA-Express adapter using the QDIO Hardware Facility.

device_addr

A hexadecimal device address that specifies the first of three consecutive virtual device numbers to be grouped for the OSA-Express adapter. TCP/IP uses *device_addr* and *device_addr+1* and *device_addr+2*.

PORTNAME port_name

The eight character name used to identify the OSA-Express adapter. If not specified, no portname will be sent to the OSA-Express adapter. See the [“Usage Notes” on page 542](#) for details.

PORTNumber nn

This operand maps a port number to the specified OSD device. If not defined, the PORTNumber defaults to 0. Specify a number in the range 0-15. The value of the port number depends on how many ports the OSA-Express adapter supports.

NONRouter

If a datagram is received at this device for an unknown IPv4 IP address, the datagram will be discarded instead of rerouted.

PRIRouter

If a datagram is received at this device for an unknown IPv4 IP address, the datagram will be routed to the TCP/IP instance to which this device was defined with the PRIRouter parameter.

SECRouter

If a datagram is received at this device for an unknown IPv4 IP address, and the TCP/IP connection defined as the primary router is not active, then the datagram will be routed to the TCP/IP instance to which this device was defined with the SECRouter parameter.

IGNORESETRoutererrors

Indicates to TCP/IP that any error which is encountered when attempting to set itself as the primary or secondary router for this device should not prevent the device from coming online. The default is to not bring a device online if TCPIP cannot honor a PRIROUTER or SECROUTER request, but when this option is specified, the device will be brought online in NONROUTER mode.

NOAUTORestart

Instructs the TCP/IP server to not attempt to restart the device.

AUTORestart

Instructs the TCP/IP server to attempt to restart the device in the event of a device failure. AUTORestart will only be attempted after successful data transfer has occurred. By default, TCP/IP restarts devices that have failed.

IPV6NONRouter

If a datagram is received at this device for an unknown IPv6 address, the datagram will be discarded instead of rerouted. This option applies to the QDIOETHERNET link type only and will be ignored if specified for other link types.

IPV6PRIRouter

If a datagram is received at this device for an unknown IPv6 address, the datagram will be routed to the TCP/IP instance to which this device was defined with the IPV6PRIRouter parameter. This option applies to the QDIOETHERNET link type only and will be ignored if specified for other link types.

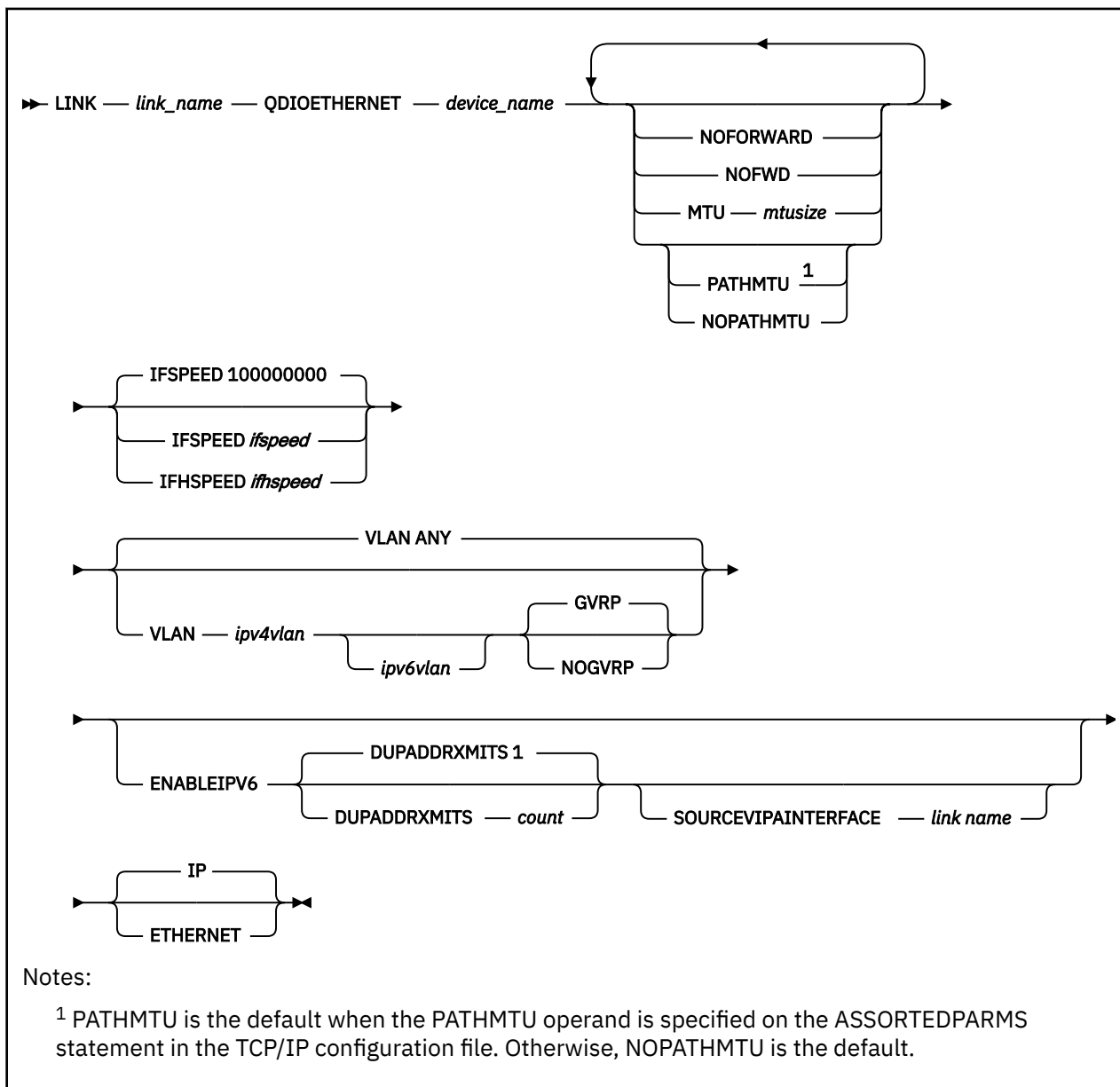
IPV6SECRouter

If a datagram is received at this device for an unknown IPv6 address, and the TCP/IP connection defined as the primary router is not active, then the datagram will be routed to the TCP/IP instance to which this device was defined with the IPV6SECRouter parameter. This option applies to the QDIOETHERNET link type only and will be ignored if specified for other link types.

Usage Notes

- TCP/IP assigns roles to the three OSA device addresses starting with *device_addr*. The *device_addr* is the data device. *device_addr_1* and *device_addr_2* are the read and write devices. Make sure the read and write real device addresses are consecutive.

LINK Statement for QDIOETHERNET



Operands

link_name

A unique name for the link. The maximum length is 16 characters. Note that numeric interface names containing a decimal point (123.456, for example) cannot be used when running MPROUTE.

QDIOETHERNET

Specifies that the link is an Ethernet connection on a Queued-Direct-I/O (QDIO) adapter.

device_name

The *device_name* must be the one that was specified in the DEVICE statement.

NOFORWARD

NOFWD

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtusize* are allowed:

- Zero, which indicates TCP/IP should use an intelligent default value based on the link type. See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.
- From 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

Note: The IPv6 intelligent default MTU value is 1500 bytes. When a LINK is enabled for IPv6 and the *mtusize* value specified is less than 1280 bytes, the IPv6 MTU value used for the link is set to 1280 bytes.

This operand overrides MTU values in the following:

- GATEWAY statement
- MPROUTE CONFIG file
- IPv6 router advertisements.

Note: IPv6 neighbor discovery routes learned by router advertisements use the LINK MTU *mtusize* before using the MTU option in an advertisement. If neither are available, the value 1280 is used.

If you specify the LINK MTU operand, IBM recommends coding 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not code 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtusize* according to information returned to TCP/IP by the device. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtusize*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was returned to TCP/IP by the device. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG.

To change the value of *mtusize*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtusize* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information on intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU**NOPATHMTU**

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default. These operands have no effect on IPv6 routes. Path MTU discovery is always enabled for IPv6 and cannot be disabled.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for the *ifspeed* variable for a QDIOTHERNET link is 0; the maximum value is 2147483647 bits per second. The default is 100000000 bits per second.

Until the interface is started successfully, the value is used by SNMP as the value of the ifSpeed MIB object. After the interface is started successfully, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

IFHSPEED *ifhspeed*

An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for the *ifhspeed* variable for a QDIOTHERNET link is 0; the maximum value is 2147 million bits per second. The default is 100 million bits per second.

Until the interface is started successfully, the value is used by SNMP as the value of the ifHighSpeed MIB object. After the interface is started successfully, SNMP uses the actual speed reported by the interface as the value of the ifHighSpeed MIB object. The value of this parameter has no effect on the operation of the device.

VLAN ANY

Specifies that no identifier for a Virtual Local Area Network (VLAN) is to be associated with this interface. This is the default.

VLAN *ipv4vlan* [*ipv6vlan*]

Specifies the identifier for one or two Virtual Local Area Network (VLAN) identifiers. *ipv4vlan* and *ipv6vlan* are numbers from 1 to 4094.

Do not use the VLAN operand to specify VLAN identifiers when the link is connected to a switch port that is configured as an access port.

If both the VLAN and ENABLEIPv6 options are used but *ipv6vlan* is not specified, *ipv6vlan* defaults to the value for *ipv4vlan*.

ipv6vlan is ignored when **ENABLEIPv6** is not specified.

GVRP

indicates that the VLAN ID should be registered with GVRP-aware switches on the LAN. GVRP provides dynamic VLAN registration and VLAN registration removal for networking switches. This eliminates the need to manually configure the individual port VLAN assignments.

NOGVRP

Do not register VLAN IDs with GVRP-aware switches on the LAN.

ENABLEIPv6

Specifies that the link should allow IPv6 traffic.

DUPADDRXMTS

Specifies how many times the link should attempt duplicate address detection on the IPv6 addresses associated with the link.

count

Specifies the number of times to attempt duplicate address detection. Specify count as an integer in the range of 0 through 255. If you specify zero, duplicate address detection will not be performed. The default is 1.

SOURCEVIPAINTERFACE *link name*

Specifies the LINK name of the virtual interface whose address(s) are to be used as the source IP address for outgoing datagrams on this link. The interface specified must be a virtual interface which has been enabled for IPv6. If the interface specified has multiple IP addresses defined, the source address will be selected from among those addresses using the default source address selection algorithm.

IP

ETHERNET

indicates whether the transport for the link is ETHERNET or IP. An ETHERNET link operates at the Layer 2 level of the OSI model while an IP link operates at Layer 3. For Layer 2 mode, each link uses a locally defined MAC address which can be generated automatically or configured through CP. For more information, see [Media Access Control \(MAC\) Address in z/VM: Connectivity](#).

Usage Notes

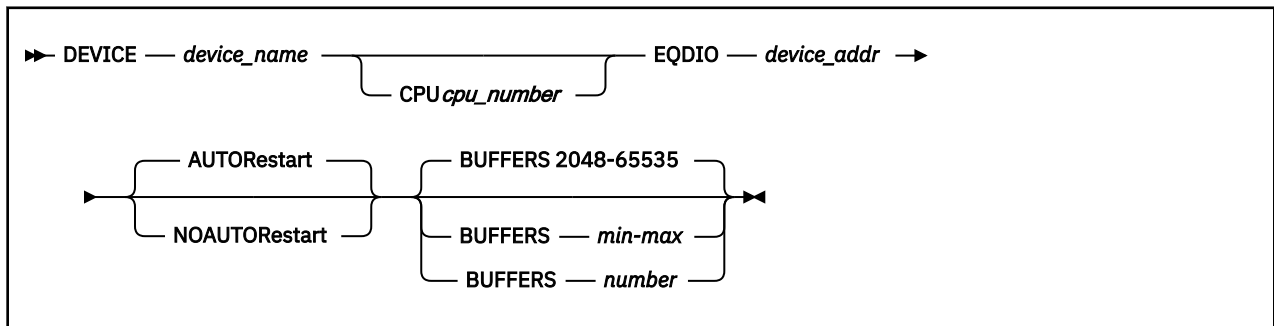
- For QDIOETHERNET links, if you specify the ENABLEIPV6 option and if the physical OSD hardware device type does not support IPv6 protocol, the device operates according to IPv4 protocol, ignoring any IPv6-related options for this link.
- VLAN configuration information can be found in [Planning for Guest LANs and Virtual Switches in z/VM: Connectivity](#).

DEVICE and LINK statements for OSH devices

Note: You can use OSD or OSH statements to configure a TCP/IP EQDIO connection.

DEVICE Statement – OSH devices

Use the DEVICE statement to specify the name and address of the device that will use the Enhanced Queued Direct I/O (EQDIO) Hardware Facility.



Operands

device_name

Specifies a unique name for the device. The maximum length is 16 characters. This name is referenced in the LINK statement.

CPU cpu_number

Specifies an integer between 0 and 6 that designates the virtual processor used to run the device driver for the associated device.

EQDIO

Specifies that the device uses the Enhanced Queued Direct I/O (EQDIO) Hardware Facility.

device_addr

Specifies a hexadecimal device address that specifies the virtual device number to be used with the EQDIO adapter.

AUTORestart

Instructs the TCP/IP server to try restarting the device in the event of a device failure. AUTORestart is attempted only after successful data transfer has occurred. By default, TCP/IP restarts devices that have failed.

NOAUTORestart

Instructs the TCP/IP server to not attempt to restart the device.

BUFFERS *min-max*

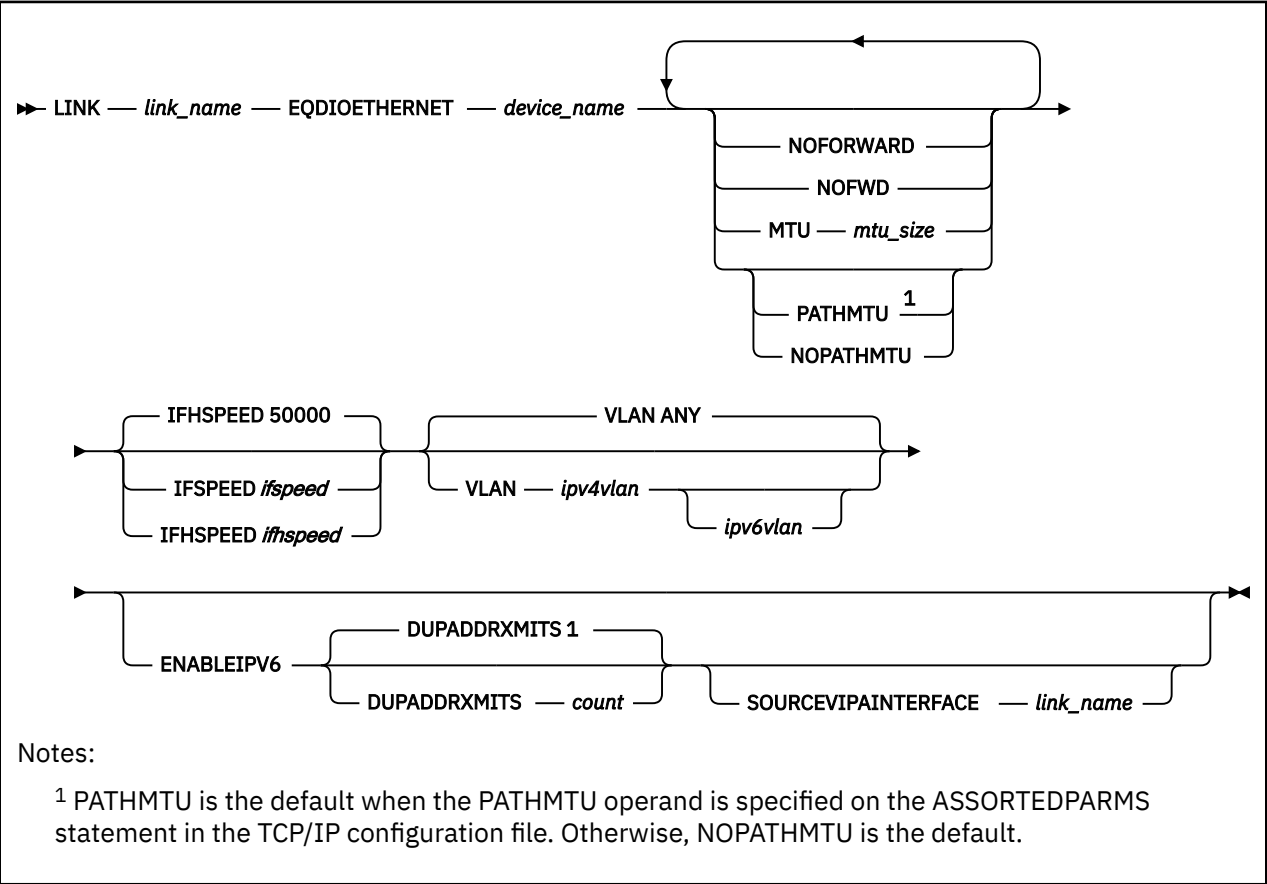
BUFFERS *number*

Specifies a number or a minimum and maximum number of buffers to be used by an EQDIO device. The number of buffers can be between 2048 and 65535. If not specified, the number of buffers defaults to a minimum of 2048 buffers and a maximum of 65535 buffers.

For a range, specify two numbers separated by a single hyphen (-) with no blank spaces — for example: x-y, where x is less than y.

Note: Due to internal device architecture, *minbuffers*, *maxbuffers*, or both values might be rounded to an approximate value. For example, if 3000-65000 is specified, the *minbuffers* and *maxbuffers* values implemented might be 3008-64992. Likewise, a single value such as 50000 might result in a value of 49984 being used.

LINK statement for EQDIOETHERNET



Operands

link_name

Specifies a unique name for the link. The maximum length is 16 characters.

Note: Numeric interface names that contain a decimal point (123.456, for example) cannot be used when running MPROUTE.

EQDIOETHERNET

Specifies that the link is an Ethernet connection on an Enhanced Queued Direct I/O (EQDIO) adapter.

device_name

Indicates the device name that was specified in the DEVICE statement.

NOFORWARD**NOFWD**

Specifies that packets received on this link will not be forwarded to another host (that is, packets destined for a foreign host will be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link will be dropped, as will packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

MTU *mtu_size*

Specifies the maximum transmission unit (MTU) size in bytes to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

The following values for *mtu_size* are allowed:

- A value of 0, which indicates that TCP/IP should use an intelligent default value based on the link type.

See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

- A value from 576 to the value specified on the LARGEENVELOPEPOOLSIZE statement.

Note: The IPv6 intelligent default MTU value is 1500 bytes. When a LINK is enabled for IPv6 and the *mtu_size* value specified is less than 1280 bytes, the IPv6 MTU value used for the link is set to 1280 bytes.

This operand overrides MTU values in the following:

- The GATEWAY statement.
- The MPROUTE CONFIG file.
- IPv6 router advertisements. IPv6 neighbor discovery routes learned by router advertisements use the LINK MTU *mtu_size* before using the MTU option in an advertisement. If neither are available, the value 1280 is used.

If you specify the LINK MTU operand, IBM recommends that you use 0 for the MTU value on the GATEWAY statement and in the MPROUTE CONFIG file. If you do not use 0 and the values differ with the LINK MTU operand, a warning message is displayed.

If you do not specify the LINK MTU option, TCP/IP assigns the ifMtu value in TCP/IP MONITOR records and the *mtu_size* according to information returned to TCP/IP by the device. However, when routes are added to TCP/IP, TCP/IP uses the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To check the value of *mtu_size*, use the NETSTAT DEVLINKS command or check the ifMtu value in the TCP/IP APPLMON MONITOR records. NETSTAT DEVLINKS shows the MTU size associated with the link that was specified as an MTU operand on the LINK statement or was returned to TCP/IP by the device. If the MTU option is not specified on the LINK statement, the MTU size shown by NETSTAT DEVLINKS might not match the MTU values specified on the GATEWAY statement or in the MPROUTE CONFIG file.

To change the value of *mtu_size*, issue an OBEYFILE or NETSTAT OBEY command and use the prior LINK statement with the *mtu_size* setting or use the IFCONFIG command. The only other LINK statement operands that can be changed are the PATHMTU and NOPATHMTU operands. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

When path MTU discovery is enabled, the MTU specified on the LINK statement is used as the starting MTU. If the MTU operand is not configured on the LINK statement, the intelligent default MTU for the device is used. For more information about intelligent default MTU values, see [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

PATHMTU**NOPATHMTU**

Specifies the use of path MTU discovery on IPv4 routes for the given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP

configuration file. Otherwise, NOPATHMTU is the default. These operands have no effect on IPv6 routes. Path MTU discovery is always enabled for IPv6 and cannot be disabled.

To enable or disable path MTU discovery, issue an OBEYFILE or NETSTAT OBEY command and then use the prior LINK statement with the path MTU discovery setting or use the IFCONFIG command. The only other LINK statement operand that can be changed is the MTU operand. A warning message will be issued if operands other than MTU, PATHMTU, or NOPATHMTU are specified.

IFSPEED *ifspeed*

Specifies an optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for the *ifspeed* variable for an EQDIOETHERNET link is 0; the maximum value is 2147483647 bits per second. The default for EQDIOETHERNET links is expressed using IFHSPEED.

Until the interface is started successfully, the value is used by SNMP as the value of the ifSpeed MIB object. After the interface is started successfully, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

IFHSPEED *ifhspeed*

Specifies an optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for the *ifhspeed* variable for an EQDIOETHERNET link is 0; the maximum value is 2147483647 million bits per second. The default is 50000 million bits per second.

Until the interface is started successfully, the value is used by SNMP as the value of the ifSpeed MIB object. After the interface is started successfully, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

VLAN ANY

Specifies that no identifier for a virtual local area network (VLAN) is to be associated with this interface. This is the default.

VLAN *ipv4vlan* [*ipv6vlan*]

Specifies the identifier for one or two VLAN identifiers. *ipv4vlan* and *ipv6vlan* are values from 1 to 4094.

Do not use the VLAN operand to specify VLAN identifiers when the link is connected to a switch port that is configured as an access port.

If both the VLAN and ENABLEIPv6 options are used but *ipv6vlan* is not specified, *ipv6vlan* defaults to the value for *ipv4vlan*.

When ENABLEIPv6 is not specified, *ipv6vlan* is ignored.

ENABLEIPv6

Specifies that the link should allow IPv6 traffic.

DUPADDRXMTS

Specifies how many times the link should attempt duplicate address detection on the IPv6 addresses associated with the link.

count

Specifies the number of times to attempt duplicate address detection. Specify *count* as an integer in the range of 0 through 255. If you specify 0, duplicate address detection will not be performed. The default is 1.

SOURCEVIPINTERFACE *link_name*

Specifies the link name of the virtual interface with one or more addresses that are to be used as the source IP address for outgoing datagrams on this link. The interface specified must be a virtual interface that has been enabled for IPv6. If the interface specified has multiple IP addresses defined, the source address is selected from among those addresses using the default source address selection algorithm.

DEVICE and LINK statements for virtual IP addressing (VIPA) devices

DEVICE Statement — Virtual Devices (VIPA)

Use the DEVICE statement to specify the name and virtual address of a device.

```
➤➤ DEVICE — device_name — CPU cpu_number — VIRTUAL — 0 —➤➤
```

Operands

device_name

A unique name for the device. The maximum length is 16 characters. This name is referenced in the LINK statement.

CPU cpu_number

Specify an integer between 0 and 6 that designates the base processor, which must be used to run the device driver for the associated device.

VIRTUAL

Specifies that this device is not associated with real hardware and is used to provide fault tolerance. Virtual devices always stay active and are never subject to physical failure.

0

A constant.

LINK Statement — Virtual Devices (VIPA)

Use the LINK statement to define the link that corresponds to a virtual device.

```
➤➤ LINK — link_name — VIRTUAL — 0 — device_name — ENABLEIPV6 —➤➤
```

Operands

link_name

A unique name for the link. The maximum length is 16 characters. The same name is specified in the HOME statement. Note that numeric interface names containing a decimal point (for example, 123.456) cannot be used when running MPROUTE.

VIRTUAL

Specifies that the link is virtual and is not associated with real hardware. It is used for fault tolerance support.

0

A constant.

device_name

The device name must be the one specified on the DEVICE statement.

ENABLEIPV6

Specifies this interface is an IPv6 VIPA interface.

Note: IPv6 virtual IP addresses cannot be auto-configured. They must be defined on the HOME statement.

Examples

The following is an example of DEVICE and LINK statements for virtual devices (VIPA).

```

DEVICE VDEV1  VIRTUAL 0
LINK   VLINK1 VIRTUAL 0 VDEV1
DEVICE VDEV2  VIRTUAL 1
LINK   VLINK2 VIRTUAL 0 VDEV2

```

Usage Notes

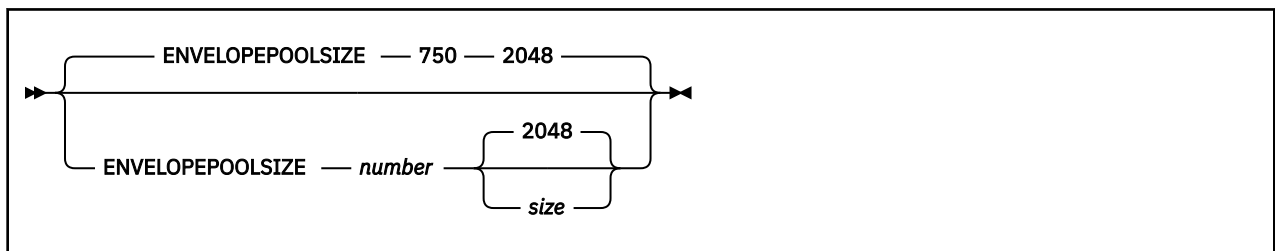
- The term *virtual device* is used to describe a VIPA device; it is in no way related to VM's traditional virtual device support.
- Only one virtual link can be defined for a virtual device.
- More than one virtual DEVICE/LINK pair can be defined to provide multiple virtual IP addresses for a TCP/IP image.
- A virtual LINK cannot be coded on the START, GATEWAY, or TRANSLATE statements.
- For rules about virtual IP address definitions for virtual links, see [“HOME Statement” on page 579](#).

More Information

- [“HOME Statement” on page 579](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)
- [“Point-to-point connections to other hosts” on page 520](#)

ENVELOPEPOOLSIZE Statement

Use the ENVELOPEPOOLSIZE statement to set the initial number and size of envelopes. Envelopes are used to maintain datagrams and fragments of up to a size of 65 535 bytes of data during TCP/IP processing.



Operands

number

The initial number of envelopes in the free pool. The default is 750.

size

The size of each envelope (in bytes); this *size* determines the maximum number of bytes that can be held by an envelope. The default is 2048. Only the following values or their alternatives (shown in parenthesis) may be specified:

```

512
1024 (1K)
2048 (2K)
4096 (4K)
8192 (8K)
16384 (16K)
32768 (32K)
65535 (64K)

```


Examples

The following example shows an ENVELOPEPOOLSIZE statement that defines the number of envelopes to be the default of 750.

```
EnvelopePoolSize 750
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- Specify the size operand to establish the size of the largest packet that can be sent and received.

Note: Ensure the specified *size* does not exceed that defined for large envelopes (defined by the LARGEENVELOPEPOOLSIZE statement). Failure to do so will result in TCP/IP stack initialization errors.

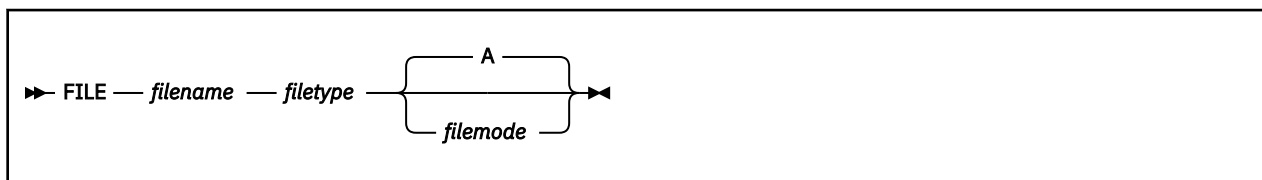
- The system will attempt to dynamically allocate 10% more envelopes any time the envelope free pool level drops below 5%. You can use the NETSTAT POOLSIZE command to monitor how many small envelopes your system is using. To avoid dynamic allocation of small envelopes during operation, use the ENVELOPEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- Running out of envelopes will result in lost packets.

More Information

- “LARGEENVELOPEPOOLSIZE Statement” on page 591
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

FILE Statement

Use the FILE statement to specify a file to receive trace information.



Operands

filename

The CMS file name of the file to receive the trace information.

filetype

The CMS file type of the file to receive the trace information.

filemode

The CMS file mode of the file to receive the trace information.

Usage Notes

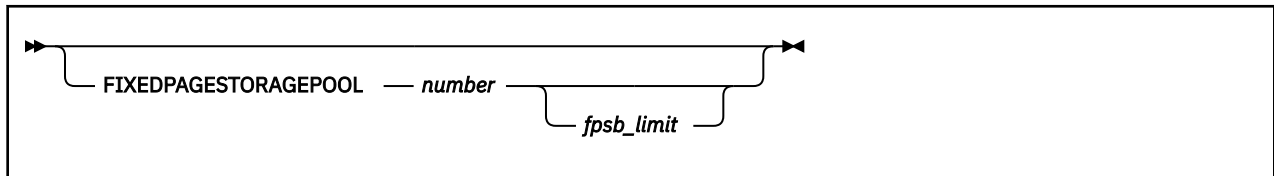
- The current trace file on disk is closed and a new file is opened as the trace file. To access the trace data, log onto the virtual machine running the TCP/IP stack and open the specified file.
- If the specified trace file already exists, its previous contents are deleted.
- If tracing is currently set to the console, and a FILE statement is encountered, the console is not closed. If tracing switches back to the console later (for example, through the SCREEN statement or NETSTAT OBEY SCREEN command), the system appends trace data to the original output in the console.

More Information

- [“NOSCREEN Statement” on page 596](#)
- [“TRACE Statement” on page 623](#)

FIXEDPAGESTORAGEPOOL

Use the FIXEDPAGESTORAGEPOOL statement to set the initial number of Fixed Page Storage Blocks (FPSBs) that will be preallocated and placed in this pool during initialization. FPSBs (*number*) are 4k blocks (pages) of storage used by TCP/IP and the Queued Direct I/O Hardware Facility.



Operands

number

The initial number of FPSBs to be pre-allocated in the pool. If *number* is specified as zero or the FIXEDPAGESTORAGEPOOL statement is omitted, no storage will be pre-allocated. The FIXEDPAGESTORAGEPOOL will be populated "as needed" based on network traffic, until the maximum allowable number (*fpsb_limit*) of FPSBs are allocated or until 90% of available virtual machine storage has been consumed.

fpsb_limit

The maximum number of FPSBs that may be allocated for this virtual machine. This value must be equal to or greater than the value specified by *number*.

Examples

1. The following example shows a FIXEDPAGESTORAGEPOOL statement that defines a storage cap of 2000 pages (8 MB). No pages will be allocated for the storage pool at TCP/IP initialization. FPSBs are allocated, as network activity dictates, up to the specified *fpsb_limit*.

```
FIXEDPAGESTORAGEPOOL 0 2000
```

2. The following example shows a FIXEDPAGESTORAGEPOOL statement which defines 1500 pages (6 MB) to be allocated at TCP/IP initialization. Since *fpsb_limit* is not specified, the storage pool will default to using up to 90% of the available CMS storage within the TCP/IP virtual machine.

```
FIXEDPAGESTORAGEPOOL 1500
```

Usage Notes

- By default, the TCP/IP server manages the storage pool to utilize up to 90% of the available CMS storage within the TCPIP virtual machine. If this default does not meet your needs or you require a specific amount of storage to be allocated at TCP/IP initialization, specify your required storage allocation by using the FIXEDPAGESTORAGEPOOL statement. For each DEVICE OSD statement that is active, 1000 to 1500 FPSBs (4 to 6 MB) are recommended.
- When the number specified by the *fpsb_limit* exceeds the available CMS free storage in the TCPIP server virtual machine, TCP/IP will recalculate the FPSB_LIMIT to be the total available CMS free storage, maintaining at least 10 free pages within the CMS storage pool.

More Information

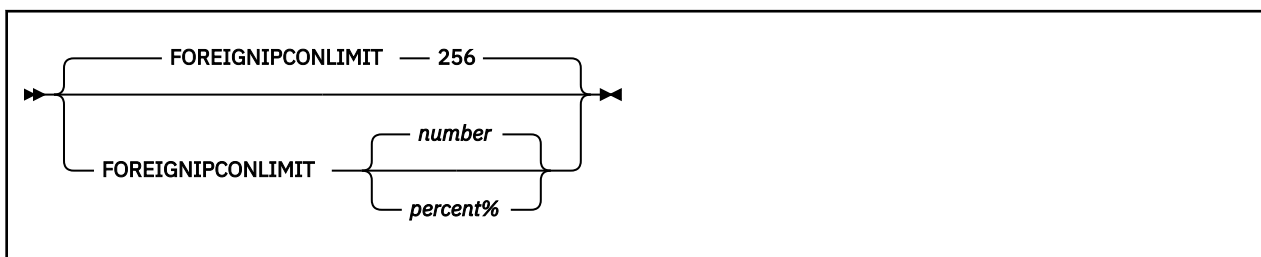
- [“DEVICE and LINK statements for OSD devices” on page 553](#)

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

FOREIGNIPCONLIMIT Statement

Use the FOREIGNIPCONLIMIT statement to define the maximum number of connections that a foreign IP address is allowed to have open at the same time. If this value would be exceeded, an SSTRESS denial-of-service attack is declared and further connection attempts from the respective IP address will be rejected until the number of open connections drops below the limit. If a value of 0 is specified, the number of connections per IP address is unlimited.

If FOREIGNIPCONLIMIT is not specified, the number of connections per IP address is limited to the initial TCBPOOLSIZE (see [“TCBPOOLSIZE Statement”](#) on page 620).



Operands

number

The maximum number of connections that a foreign IP address is allowed to have open at the same time. The default is 256 and is based on the initial TCBPOOLSIZE.

percent%

The maximum number of connections that a foreign IP address is allowed to have open at the same time, specified as a percentage of the current TCBPOOLSIZE.

Examples

1. The following example shows a FOREIGNIPCONLIMIT statement specifying the maximum number of connections that a foreign IP address is allowed to have open at the same time to be 60.

```
FOREIGNIPCONLIMIT 60
```

2. The following example shows a FOREIGNIPCONLIMIT statement specifying that no limit is applied to the number of connections that a foreign IP address has established at any given time.

```
FOREIGNIPCONLIMIT 0
```

Usage Notes

- A FOREIGNIPCONLIMIT value of 0 indicates that an unlimited number of connections for each foreign IP address is accepted.
- If no FOREIGNIPCONLIMIT statement has been provided, the number of connections per foreign IP address defaults to the initial TCB pool size (see [“TCBPOOLSIZE Statement”](#) on page 620).
- Use the NETSTAT DOS command to display the current limit.

More Information

- [“OBEY Statement”](#) on page 597
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT DOS and NETSTAT RESETDOS commands

FOREIGNIPPOOLSIZE Statement

Use the FOREIGNIPPOOLSIZE statement to set the initial number of foreign IP address control blocks. These are used to track all foreign IP addresses that have established TCP connections if a FOREIGNIPCONLIMIT statement has been used to limit connections per IP address.



Operands

number

The initial number of foreign IP address control blocks in the free pool. The default is 100.

Examples

The following example shows a FOREIGNIPPOOLSIZE statement that defines the number of foreign IP address control blocks to be 200.

```
FOREIGNIPPOOLSIZE 200
```

Usage Notes

- Each foreign IP addresses that maintains at least one established connection requires one control block.
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 10% of that number or a at least 5 control blocks. If it is successful in allocating that reduced initial pool size, initialization continues.
- The system will attempt to dynamically allocate 10% more foreign IP address control blocks any time the IP route control block free pool becomes empty.
- You can use the NETSTAT POOLSIZE command to monitor how many foreign IP address control blocks your system is using.
- The FOREIGNIPPOOLSIZE statement can only be specified as part of the configuration at initialization. It cannot be used with the NETSTAT OBEY command.
- As a rule of thumb, the value of FOREIGNIPPOOLSIZE should always be smaller than the one for TCBPOOLSIZE.

GATEWAY Statement

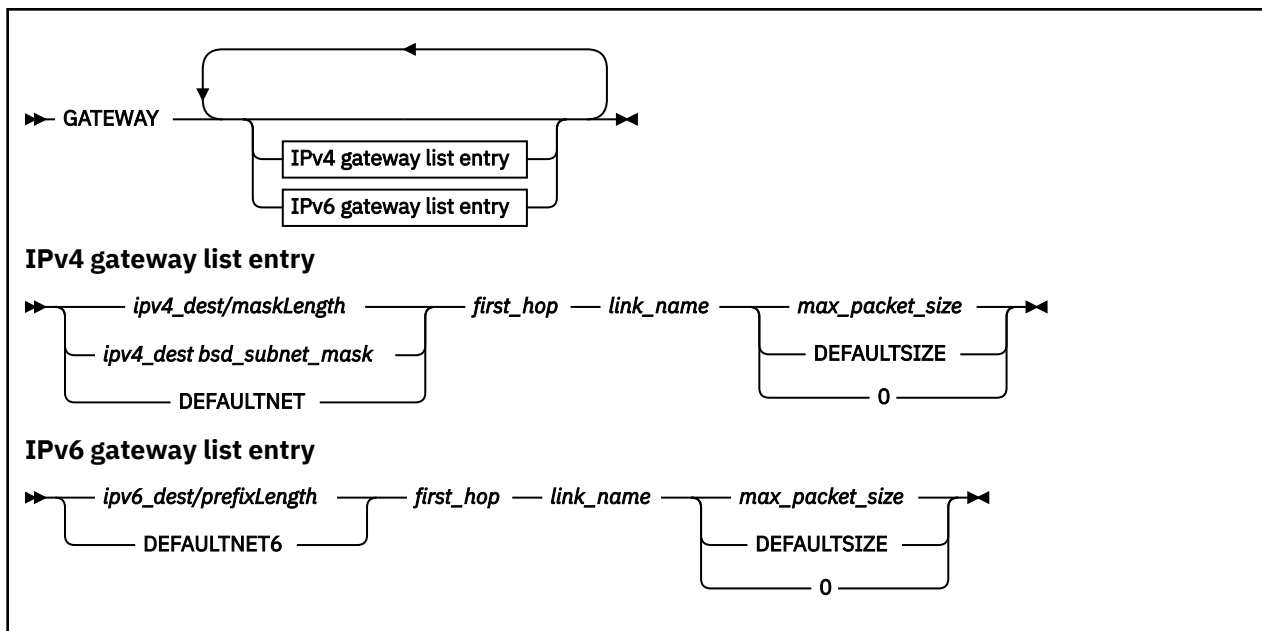
Use the GATEWAY statement to add static routes to the IP route table and to identify interfaces that are to be used with IP multicast support. Routes can be for either IPv4 or IPv6; IPv6 routes can be associated only with QDIOETHERNET or QDIOIP link types that have been enabled for IPv6.

The IP route table can be modified by any of the following:

- Replacing it using the OBEYFILE command
- Modifying the interface configuration using the IFCONFIG command
- Updating of IPv4 dynamic routes from the MPRoute server
- Accepting incoming ICMP and ICMPv6 redirect packets sent from adjacent machines, unless redirects have been disabled on the ASSORTEDPARMS statement
- Updating of IPv6 default routes from router advertisements
- Updating of IPv6 dynamic routes from the MPRoute server.

If MPRoute is running, static routes defined by the GATEWAY statement are not managed or updated by MPRoute. To have MPRoute manage these routes, remove them from the GATEWAY statement — MPRoute will find out about them dynamically. A simple way to flush the static routes from the IP routing table is to include a GATEWAY statement with no routing information in an OBEYFILE file. For further explanations of using the GATEWAY statement with MPRoute, see [Chapter 8, “Configuring the MPRoute Server,”](#) on page 193.

The first GATEWAY statement of each configuration file processed replaces the existing routing table with the new gateway information. All static routes are deleted, along with all routes learned by way of ICMP or ICMPv6 redirects. Routes created by MPRoute, along with IPv6 routes created by router advertisements are not deleted. Subsequent GATEWAY statements in the same file add entries to the routing table.



Operands

ipv4_dest

The subnet address, in dotted-decimal form, of the destination described by this route. In the event of a network that is not subnetted, *ipv4_dest* is the network address. Alternatively, *ipv4_dest* can be specified as the IP address, in dotted-decimal form, of any host that is on the destination network. Trailing zero octets can be omitted.

maskLength

An integer value in the range 1-32 specifying how many of the leftmost contiguous bits of the address specified by *ipv4_dest* comprise the subnet mask.

bsd_subnet_mask

A bit mask, expressed in dotted-decimal form, that shows the bits of network and host fields that make up the subnet. The bits describing each of the fields must be contiguous.

If this is a host entry, specify HOST instead of a *bsd_subnet_mask*.

For a supernet route, which can be used to represent multiple network routes, specify a supernet mask. The supernet mask should be specified in BSD form.

For example, the supernet 130.200 with supernet mask of 255.252.0.0 represents networks 130.200, 130.201, 130.202, and 130.203 based on the unmasked bits in the network portion of the address.

The topics "Subnetting" and "Simplified IP Datagram Routing Algorithm with Subnets" in [z/VM: TCP/IP Diagnosis Guide](#) illustrate the concept of subnetting and provide an example that describes how a subnet route is resolved. Subnets and subnet masks are also discussed in the IBM: Redbook - *IP Network Design Guide*, SG24-2580. See [Redbooks \(https://www.ibm.com/redbooks\)](https://www.ibm.com/redbooks).

DEFAULTNET

Specifies an IPv4 default to use for any network that is not explicitly routed.

You must specify a GATEWAY statement that defines the route to the *first_hop* for the DEFAULTNET keyword to take effect.

ipv6_dest

The destination IPv6 host or network, specified using prefix notation. No spaces are allowed between *ipv6_dest*, /, and *prefixLength*.

For a host route, specify *ipv6_dest*/**128**.

The destination address can be an IPv4-compatible IPv6 address. An error message is displayed if the destination address is an IPv4-mapped IPv6 address.

prefixLength

An integer value in the range 1-128 specifying how many of the leftmost contiguous bits of the address specified by *ipv6_dest* comprise the prefix.

DEFAULTNET6

Specifies an IPv6 default to use for any network that is not explicitly routed.

You must specify a GATEWAY statement that defines the route to the *first_hop* for the DEFAULTNET6 keyword to take effect.

first_hop

Specify one of the following:

- An equal sign (=), meaning that messages are routed directly to destinations on that network or directly to that host. The equal sign is not supported for DEFAULTNET or DEFAULTNET6.

Rule: For DEFAULTNET or DEFAULTNET6, you cannot specify an equal sign as a default gateway.

- The IPv4 or IPv6 address of a gateway or router that you can reach directly and that forwards messages to the destination network or host.

The destination and first hop IP addresses must both be either IPv4 or IPv6, otherwise an error message is displayed. Additionally, an error message is displayed if the first hop IP address is IPv6 and it is an IPv4-compatible IPv6 address or an IPv4-mapped IPv6 address.

link_name

The name of the link through which packets are sent to the specified network. The link name must be defined in a LINK statement and cannot be the name of a virtual link.

If you code an IPv6 address when the link specified as *link_name* is configured for IPv4 only, an error message is displayed.

max_packet_size

The maximum transmission unit (MTU) in bytes for the destination.

This value must be at least 576 and can be up to 65 535 but cannot be larger than the value specified on the LARGEENVELOPEPOOLSIZE statement. MTU values specified on the LINK statement override values specified on the GATEWAY statement, so specify a value of 0 in this field if the LINK statement for *link_name* specifies an MTU *mtusize* operand.

The DEFAULTSIZE keyword in this field requests that TCP/IP supply a value of 576 for IPv4 routes and 1280 for IPv6 routes.

IBM recommends that you use the following sizes instead of DEFAULTSIZE as the packet size for these networks:

- 1492 bytes for Ethernet 802.3
- 8992 bytes for Gigabit Ethernet
- 1500 bytes for Ethernet Version 2 IEEE
- 2048, 4352, or 4096 bytes for FDDI
- 32760 bytes for CTC
- 32764 bytes for IUCV

Maximum frame size minus 8K for HiperSockets

If any bridge or router does not perform IP-layer fragmentation of packets, you must select an MTU corresponding to the smallest MTU in use by that bridge or router. Selecting an MTU size that is too large may cause client applications to hang.

When coding equal-cost routes, use the same packet size among all routes to a single destination. If any packet size does not match the preceding ones, the packet size of the route is changed to the smaller packet size.

Examples

1. The following is an example that shows a GATEWAY statement that is divided by the types of routes used.

```
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; (IP) Network Subnet First Link Max. Packet
; Address Mask Hop Name Size (MTU)
; -----
;
; 130.50.10.0 255.255.255.0 = LINK1 2000
; 193.5.2.0 255.255.255.0 = LINK2 1500
; 128.240.5.0 255.255.255.0 = LINK3 1500
; 9.67.43.0 255.255.255.0 = LINK4 4000
; 193.7.2.2 HOST = LINK5 2000
;
; Indirect Routes - Routes that are reachable through routers on my
; network.
;
; (IP) Network Subnet First Link Max. Packet
; Address Mask Hop Name Size (MTU)
; -----
;
; 193.12.2.0 255.255.255.0 130.50.10.1 LINK1 2000
; 10.5.6.4 HOST 193.5.2.10 LINK2 1500
;
; Default Route - All packets to an unknown destination are routed
; through this route.
;
; (IP) Network Subnet First Link Max. Packet
; Address Mask Hop Name Size (MTU)
; -----
;
; DEFAULTNET 9.67.43.1 LINK4 DEFAULTSIZE
```

Figure 12. Example of route types

2. The following is an example of a network that employs variable subnets and supernets.

Figure 13 on page 572 shows a host, VM1, directly connected to variable subnets 10.2 and 10.2.8. VM1 is indirectly connected to variable subnets 121.2.3.128 and 121.2.4. In addition, VM1 is directly connected to supernet 192.3.200 and indirectly connected to supernet 130.200.

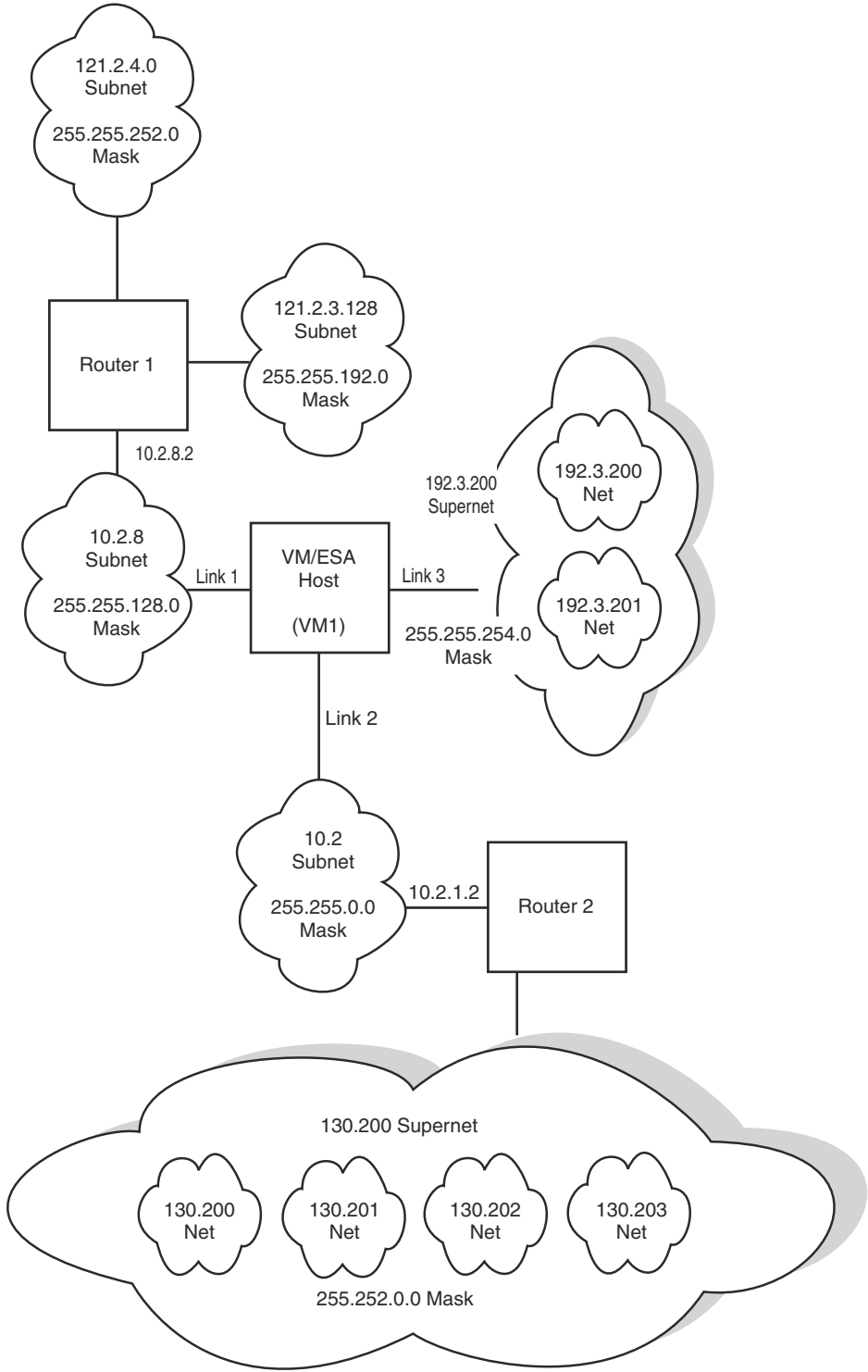


Figure 13. Example of Network Connectivity Using Variable Subnetting

The following is an example of the corresponding GATEWAY statement:

```
GATEWAY
;
; Direct and indirect subnets
;
; (IP) Network Subnet First Link Max. Packet
; Address Mask Hop Name Size (MTU)
```



```

; -----
;
; 10.2.8.0      255.255.128.0  =      LINK1      1500
; 10.2.0.0      255.255.0.0    =      LINK2      2000
; 121.2.3.128   255.255.255.192 10.2.8.2  LINK1      2000
; 121.2.4.0     255.255.252.0   10.2.8.2  LINK1      2000
;
;
; Direct and indirect supernets
;
; (IP) Network Subnet      First      Link      Max. Packet
; Address      Mask        Hop        Name      Size (MTU)
; -----
;
; 192.3.200.0   255.255.254.0  =      LINK3      2000
; 130.200.0.0   255.252.0.0    10.2.1.2 LINK2      2000

```

Network 10 has variable subnets 10.2 and 10.2.8. Subnet 10.2 uses the high-order byte of the host field as the subnet field, and subnet 10.2.8 uses the first two bytes of the host field as the subnet fields.

Network 121.2 has variable subnets 121.2.3.128 and 121.2.4, and is an example of indirect routing. For the network shown in [Figure 13 on page 572](#), IP datagrams will be delivered to the directly connected router at address 10.2.8.2, which represents the "first-hop" toward either of two destination subnets, 121.2.4 or 121.2.3.128.

Networks 130.200 and 192.3.200 are supernets that represent multiple networks. These use the high-order bytes of the network field as supernet fields. The number of networks is determined by the mask bits in a network field. Based on the masked bits in the network field, supernet 192.3.200, with a supernet mask of 255.255.254.0, represents networks 192.3.200 and 192.3.201. Supernet 130.200, with a supernet mask 255.252.0.0, represents networks 130.200, 130.201, 130.202, and 130.203.

3. The following example shows equal-cost multipath routes to common destinations to be used for IP traffic load splitting.

[Figure 14 on page 574](#) shows a host, VM1, directly connected to variable subnet 10.2.0.0 and indirectly connected to variable subnet 10.3.0.0. The paths to 10.2.1.3 on LINK1 and LINK2 illustrate multiple equal-cost direct host routes. The paths to subnet 10.2.0.0 on LINK1 and LINK2 illustrate multiple equal-cost direct subnetwork routes. The paths to subnet 10.3.0.0 on LINK1 (through Router 2) and on LINK3 (through Router 1) illustrate multiple equal-cost indirect subnetwork routes. In this example, if two packets were being sent to 10.3.7.7, one would go out through LINK1 and one would go out through LINK3.

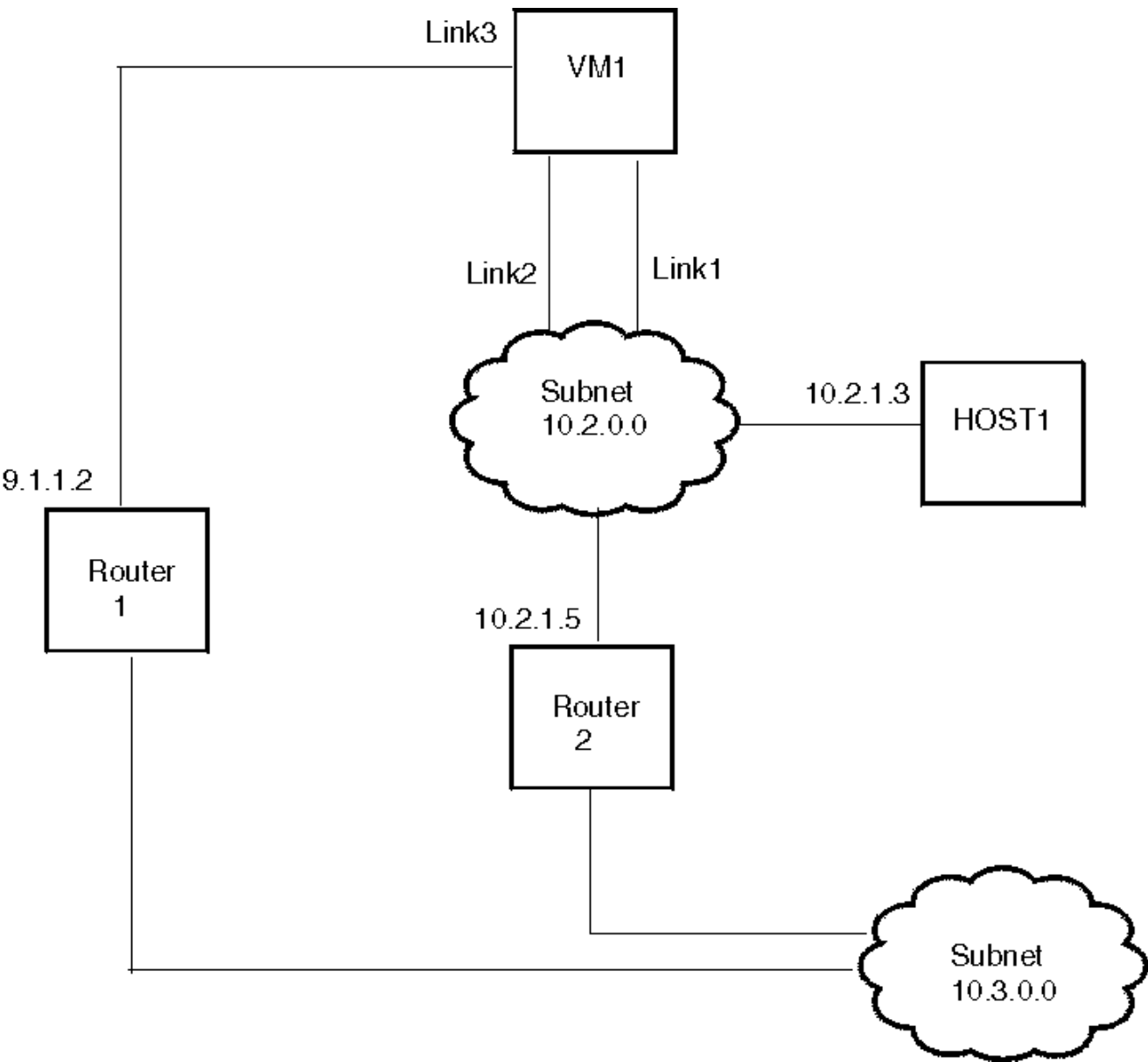


Figure 14. Example of Network Using equal-cost multipath routes

The following is an example of the corresponding GATEWAY statement:

GATEWAY					
;					
; Direct host routes					
;					
(IP) Network	Subnet	First	Link	Max. Packet	
Address	Mask	Hop	Name	Size (MTU)	
-----	-----	-----	-----	-----	
;					
10.2.1.3	HOST	=	LINK1	4000	
10.2.1.3	HOST	=	LINK2	4000	
;					
; Direct subnet routes					
;					
10.2.0.0	255.255.0.0	=	LINK1	4000	
10.2.0.0	255.255.0.0	=	LINK2	4000	
;					
; Indirect subnet routes					
;					
10.3.0.0	255.255.0.0	10.2.1.5	LINK1	1500	
10.3.0.0	255.255.0.0	9.1.1.2	LINK3	1500	

4. Consider a fictitious intranet consisting of two guest LANs (LAN2 and SUBNT240) and one Ethernet link (with access to the real Internet) as in Figure 15 on page 575.

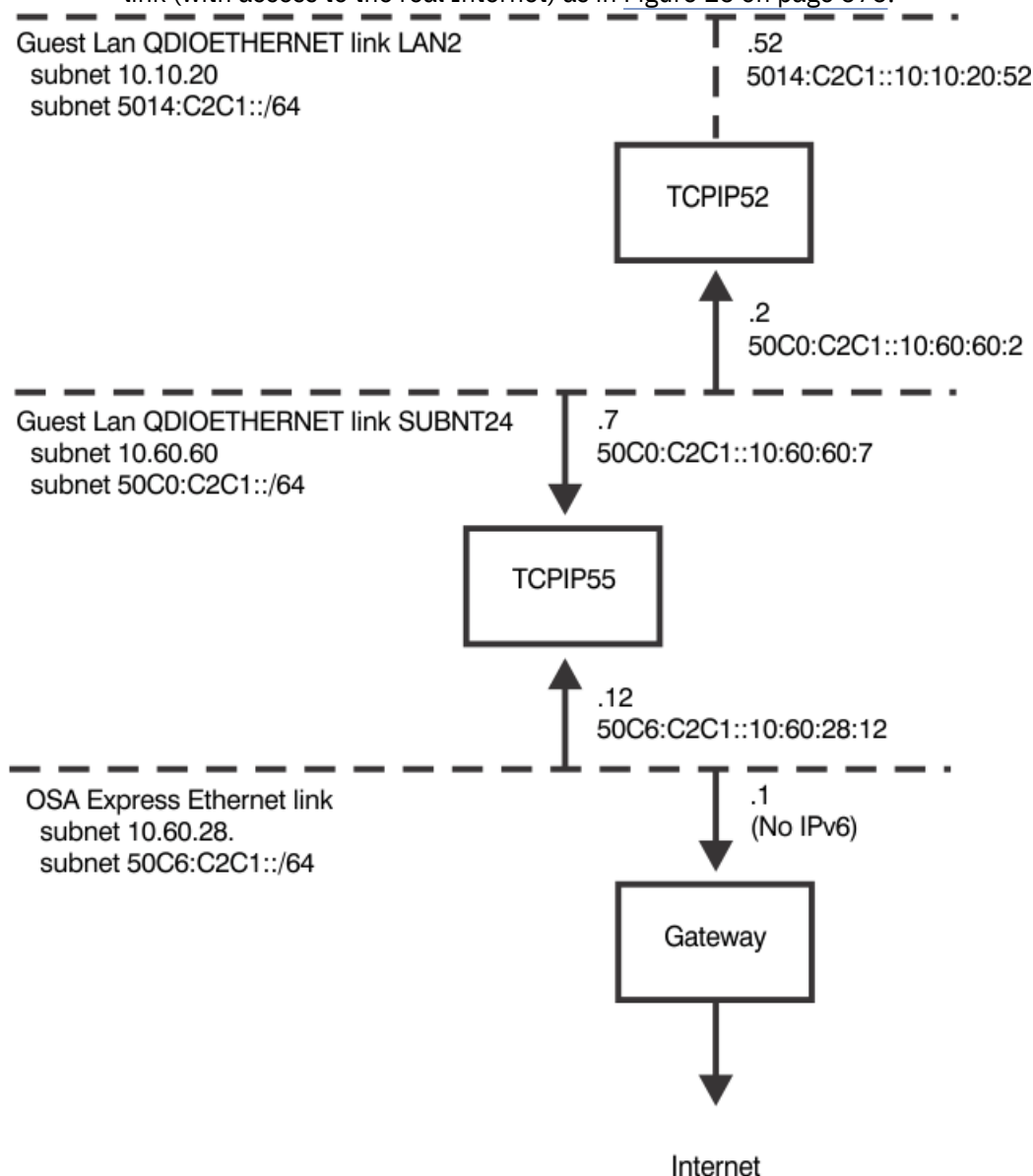


Figure 15. Intranet with Two Guest LANs

The following are examples of the DEVICE, LINK, HOME, GATEWAY and START statements for the TCP/IP for z/VM stack virtual machines TCPIP52 and TCPIP55.

For the TCPIP52 virtual machine:

```
; =====
; Start of PROFILE TCPIP for TCPIP52
; =====
...
...
; =====
; DEVICE and LINK statements
;
; Define the network interfaces used in your environment.
; =====
; Interface is on SUBNT240
DEVICE DEV5200 OSD 5200 PORTNAME TCPIP52
LINK T0240 QDIOETHERNET DEV5200 ENABLEIPV6 MTU 2000
; Interface is on LAN2
DEVICE DEV5220 OSD 5220 PORTNAME TCPIP522 PRIROUTER
LINK TOLAN2 QDIOETHERNET DEV5220 ENABLEIPV6 MTU 2000
...
```

```

...
;-----
; Define the internet (IP) address(es) for this VM host
;-----
HOME
10.60.60.2          255.255.255.0  T0240
10.10.20.52         255.255.255.0  TOLAN2
50C0:C2C1::10:60:60:2      T0240
5014:C2C1::10:10:20:52     TOLAN2
; (End of HOME address information)
...
;-----
; Static Routing Information
;
; Note: The following direct static routes will be generated based on
;       the HOME statement:
;
;       10.60.60.0/24 through interface T0240
;       10.10.20.0/24 through interface TOLAN2
;       50C0:C2C1::/64 through interface T0240
;       5014:C2C1::/64 through interface TOLAN2
;-----
GATEWAY
;
; (IPv6) Network      First      Link  Max. Packet
;       Address       Hop       Name  Size (MTU)
;-----
; 50C6:C2C1::/64 50C0:C2C1::10:60:60:7  T0240    0
;
;-----
; Define The DEFAULT route used for any network not explicitly routed
; via the previous entries.
;-----
; Default Keyword      First Hop      Link Name      MTU
;-----
; DEFAULTNET          10.60.60.1      T0240          0
;
; (End of GATEWAY Static Routing information)
...
;-----
; Start all network interface devices used in this environment.
;-----
START DEV5200
START DEV5220
; =====
; End of PROFILE TCPIP for TCPIP52
; =====

For the TCPIP55 virtual machine:
; =====
; Start of PROFILE TCPIP for TCPIP55
; =====
...
;-----
; DEVICE and LINK statements
;
; Define the network interfaces used in you environment.
; =====
; Interface is on OSA-Express Ethernet Link

DEVICE DEV1B29 OSD 1B29 PORTNAME OSA14FRE PRIROUTER
LINK ROSA QDIOETHERNET DEV1B29 ENABLEIPV6 MTU 2000
; Interface is on SUBNT240
DEVICE DEV5500 CPU 1 OSD 5500 PORTNAME TCPIP55
LINK T0240 QDIOETHERNET DEV5500 ENABLEIPV6 MTU 2000
...
;-----
; Define the internet (IP) address(es) for this VM host
;-----
HOME
10.60.28.12          255.255.255.0  ROSA
50C6:C2C1::10:60:28:12      ROSA
10.60.60.7           255.255.255.0  T0240
50C0:C2C1::10:60:60:7      T0240
; (End of HOME address information)
...

```

```

; -----
; Static Routing Information
;
; Note: The following direct static routes will be generated based on
;       the HOME statement:
;
; 10.60.28.0/24 through interface ROSA
; 10.60.60.0/24 through interface T0240
; 50C0:C2C1::/64 through interface T0240
; 50C6:C2C1::/64 through interface ROSA
;
; -----
GATEWAY
; (IPv6) Network      Link  Max. Packet
; Address             First Hop Name  Size (MTU)
; -----
; 5014:C2C1::/64 50C0:C2C1::10:60:60:2 T0240  0
;
; -----
; Define The DEFAULT route used for any network not explicitly routed
; via the previous entries.
;
; Default Keyword      First Hop      Link Name      MTU
; -----
; DEFAULTNET          10.60.60.1      T0240          0
;
; (End of GATEWAY Static Routing information)
...
; Start all network interface devices used in this environment.
; -----
START DEV5500
START DEV1B29
; =====
; End of PROFILE TCPIP for TCPIP55
; =====

```

Usage Notes

- If the gateway table is empty, the loopback test addresses are still routed properly.
- If a syntax error is found in a GATEWAY statement, the remainder of the statement is ignored. Subsequent GATEWAY statements in the same profile or OBEYFILE are processed.
- Routes are used in the following sequence:
 1. If a route exists to the destination address (a host route), it is chosen first.
 2. For IPv4, if subnet, network, or supernet routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen second.
 3. For IPv6, if prefix routes exist to the destination, the route with the most specific prefix is chosen second.
 4. Default routes are chosen when no other route exists to a destination.
- An interface can be defined to send multicast datagrams. This can be accomplished by specifying the general multicast group address of 224.0.0.0 for an interface, as in this example:

```

...
GATEWAY
; (IP) Network  Subnet      First      Link      Max. Packet
; Address      Mask        Hop        Name      Size (MTU)
; -----
; 224.0.0.0    HOST        =          LINK1    DEFAULTSIZE

```

Sending interfaces can also be defined for specific multicast groups. In the example that follows, the LINK2 link entry identifies that this link is to be used to send datagrams to the 224.1.1.1 multicast address:

```

GATEWAY
; (IP) Network  Subnet      First      Link      Max. Packet
; Address      Mask        Hop        Name      Size (MTU)

```

```

; -----
224.1.1.1    HOST      =      LINK2      DEFAULTSIZE

```

The interface used for sending an outbound multicast datagram is selected based on the following precedence:

1. Use the interface associated with a specific socket, as specified with a `send()`, `sendmsg()`, or `sendto()` call for which the `MSG_DONTROUTE` parameter has also been specified.
 2. Use an application-specified interface, as determined through the `setsockopt()` call for which the `IP_MULTICAST_IF` parameter has been specified.
 3. Use an interface that is associated with a specific multicast group address, perhaps 224.1.1.1.
 4. Use an interface that is associated with the general multicast group address (224.0.0.0).
 5. Use the default network interface (identified by the GATEWAY statement `DEFAULTNET` operand), provided its associated link is multicast-capable. If this default interface is not multicast-capable, attempts to send multicast datagrams receive an `ENETUNREACH` error.
- Packet size considerations:
 - Information is transferred over a TCP connection in discrete packets. Each packet includes a TCP header and an IP header. The header size is independent of the amount of user information included, so the larger the packets sent, the less relative bandwidth is consumed by protocol headers. Also, the TCP software layer consumes a fixed amount of CPU time for each packet, independent of the packet size.
 - *max_packet_size* must be at least 576 and may not exceed the value specified for *lrg_env_size* in the `LARGEENVELOPEPOOLSIZE` statement. See [“LARGEENVELOPEPOOLSIZE Statement” on page 591](#) for more information about large envelope size. Some networks limit the packet size to a smaller value. For example, while the largest packet size for the Ethernet protocol is 1500 bytes, the largest packet size for the 802.3 protocol is 1492.
 - The actual packet size will be determined by the total network connection:
 - If a locally-attached host has a packet size smaller than yours, transfers to that host will use the smaller size.
 - Large packets can be fragmented by intervening gateways **for IPv4 only**. Fragmentation and reassembly of packets are expensive in their use of bandwidth and CPU time. To reduce these costs, packets sent through gateways to other networks should use the default size, `DEFAULTSIZE`, unless all intervening gateways and networks are known to accept larger packets. However, to minimize TCP/IP for z/VM overhead, larger packets sizes should be used. For additional information, refer to [z/VM: Performance](#).
 - If this is a Router link, then the *max_packet_size* cannot exceed the *write_size* specified on the corresponding `DEVICE` statement.
 - IPv6 static routes configured using the GATEWAY statement take precedence over routes learned from received Router Advertisements. Static routes will not be replaced by Router Advertisement routes.
 - IPv6 routes for link-local prefix or subnet routes are not allowed on the GATEWAY statement because link-local addresses are not subnetted.
 - The IPv4 *bsd_subnet_mask* must follow the Classless Inter-Domain Routing (CIDR) convention that requires the actual mask to be one or more on-bits followed by zero or more off-bits. You cannot have on-bits followed by off-bits followed by on-bits. Therefore, a class A mask of 255.255.254.0 is valid, but a class A mask of 255.255.253.0 is not valid because 253 is 11111101.
 - A general multicast default route for IPv6 can be specified using:

```

GATEWAY
;
; (IP) Network First      Link      Max. Packet
; Address      Hop       Name       Size (MTU)
; -----
; FF00::/8     =         LINKA     4096
;

```

Figure 16. An IPv6 multicast default route on the GATEWAY statement

- The GATEWAY statement overrides any generated routes that are added during HOME statement processing.
- There is no limit on the number of equal-cost multipath routes to a destination.
- If there are multiple equal-cost paths to a destination network or host, TCP/IP, upon sending an IP packet to a given host in that destination network, selects a path on a round-robin basis from a multipath routing list to that destination host. Connection-oriented packets (IPv4 and IPv6) will be in round-robin order on a per-connection basis. When a connection is opened, TCP/IP will select a path on a round-robin basis from a multipath routing list. The selected path will be used for routing all packets for that connection (as long as the path is active). When a new connection is opened, the round-robin strategy will be used to select a new path. Connectionless packets (IPv4 and IPv6) will be in round-robin order on a per-packet basis. The selected path is used for routing that IP packet. All packets for a given association with a destination host are spread among the multiple equal-cost paths.

When MPRoute is being used, multiple routes to the same destination can be dynamically added to the TCP/IP stack's routing table, based upon information learned from other routers that are configured to run the same dynamic routing protocols. These multiple routes are added when the route calculation for each results in the same route cost value (metric).

Equal-cost multipath routes have the same MTU value. You can specify the MTU value on the following:

- The GATEWAY statement
- The OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE statements in the MPRROUTE CONFIG file
- The LINK statement.

The MTU value on the LINK statement overrides the others. If you use OBEYFILE to change a LINK statement value when equal-cost multipath routes are defined, you should update the LINK statement MTU values for all associated links to the new value. Otherwise, equal-cost multipath routes are assigned either the lowest MTU value from associated LINK statements or, when there are links configured without the LINK statement MTU option, the lower of the MTU values from the associated GATEWAY statement or the MPRROUTE CONFIG file.

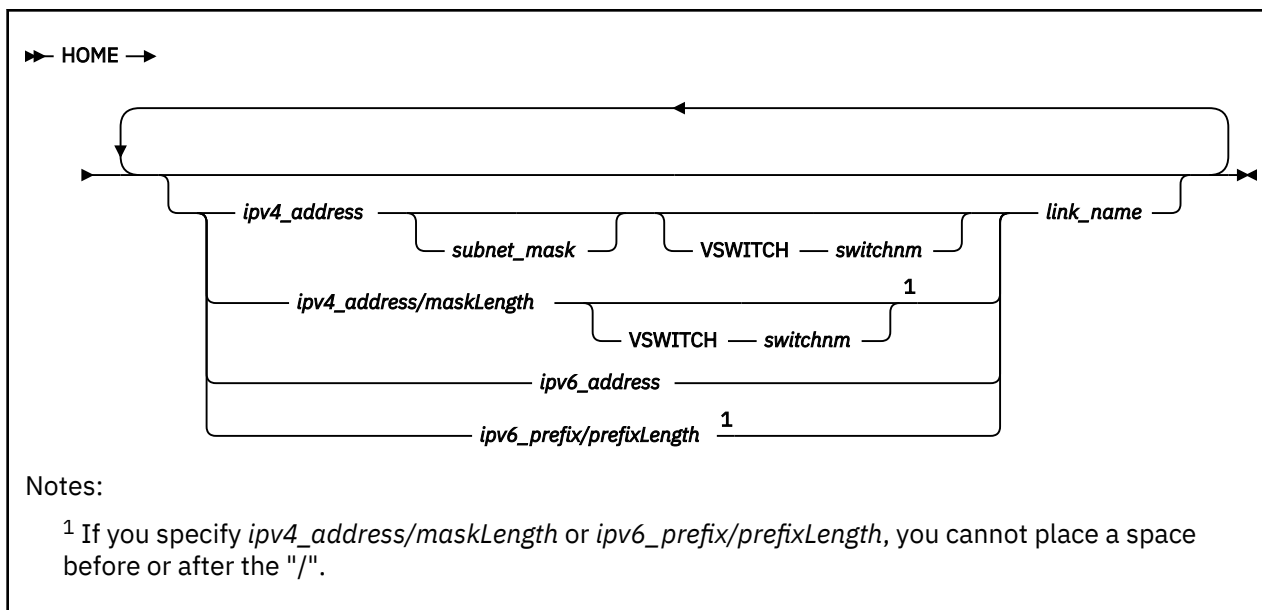
More Information

- [“ASSORTEDPARMS Statement” on page 529](#)
- [“Attaching network devices to the TCPIP virtual machine” on page 519](#)
- [“LARGEENVELOPEPOOLSIZE Statement” on page 591](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)
- [Chapter 8, “Configuring the MPRoute Server,” on page 193](#)

HOME Statement

The HOME statement defines IP addresses and their associated subnet masks and link names (called the HOME list).

The first HOME statement of each configuration file processed replaces the existing HOME list with a new one. Subsequent HOME statements in the same file add entries to the list.



Operands

ipv4_address

One of the IPv4 addresses valid for this host. The address can be associated with any type of link. It must be specified in dotted-decimal form.

subnet_mask

Specifies the subnet mask, in dotted-decimal format, associated with the IPv4 address. *subnet_mask* should be specified in BSD-style, meaning that the high order bits must be "1" (for example, 255.255.255.0, not 0.255.255.0).

Note: The subnet mask described by the subnet mask operand must be between 128.0.0.0 and 255.255.255.252. The IPv4 address must not be the subnet address or broadcast address for the network described by the IPv4 address/subnet_mask pair (that is, it cannot be the first or last address in the subnet's range). For point-to-point links, a subnet mask of 255.255.255.252 is recommended.

VSWITCH *switchnm*

The link is providing management services for the given virtual switch. For more information, see [Configuring an SNMP Subagent for a Virtual Switch in z/VM: Connectivity](#).

This keyword may be specified with IPv4 addresses only.

ipv4_address/maskLength

Is equivalent to specifying a *subnet_mask* in which the left-most *maskLength* bits are "1" (for example, `/24` is equivalent to a *subnet_mask* of 255.255.255.0).

Note: The maskLength must be between /1 and /30. The IP address must not be the subnet address or broadcast address for the network described by the IPv4 address/maskLength pair (that is, it cannot be the first or last address in the subnet's range). For point-to-point links, a maskLength of /30 is recommended.

ipv6_address

One of the IPv6 addresses valid for this host. IPv6 addresses can be associated with QDIOETHERNET, QDIOIP, and VIRTUAL link types. TCP/IP always auto-configures a link-local address for an IPv6 enabled link.

The following IPv6 addresses are not accepted for *ipv6_address*:

- Link local IP addresses
- Multicast IP addresses
- IPv4-mapped IPv6 addresses

- IPv4-compatible IPv6 addresses
- Loopback address (: : 1)
- Unspecified address (: :)

The IPv6 address can be specified in either the preferred or compressed form.

Example: Preferred form:

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

Compressed form:

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

ipv6_prefix/prefixLength

Tells TCP/IP to create an IPv6 address using the specified prefix and appending the interface ID to it. For autoconfiguration, the prefix length must be 64 (/64). A complete address may be specified using a prefix length of 128. If a prefix length other than 64 or 128 is specified, an error message is issued and rest of the HOME statement is ignored.

link_name

The name of the link (defined in a previous LINK statement) associated with the home address.

Examples

1. The following example shows a HOME statement that defines the IP addresses of each link to the host. The corresponding GATEWAY statement is shown in Example [Figure 12 on page 571](#).

```
Home
  130.50.75.1    LINK1
  193.5.2.1      LINK2
  9.67.43.110    LINK4
  193.7.2.1      LINK5
```

The above Home entries are examples of actual links that are associated with physical IP addresses.

2. For an example HOME statement in a source VIPA configuration, see [“Configuring Source VIPA” on page 516](#).
3. In the following example, QDO1 is an OSA-Express Ethernet adapter that is configured to use the Queued Direct I/O Hardware Facility. The home IPv4 address of the z/VM host is 125.0.0.33 with a subnet mask of 255.255.255.0. The home IPv6 address is FEC0::1:9:67:115:66.

```
DEVICE QDO1 OSD 1D01 PORTNAME BIGANG
LINK GIG1 QDIOETHERNET QDO1 ENABLEIPV6

HOME
125.0.0.33 255.255.255.0 GIG1
FEC0::1:9:67:115:66 GIG1
```

4. In the following example, QDO2 is an OSA-Express Ethernet adapter that is configured to use the Queued Direct I/O Hardware Facility. The home IPv4 address of the z/VM host is 125.0.0.34 with a subnet mask of 255.255.0.0. The home IPv6 prefix is FEC0:0:0:1:0:0:0/64.

```
DEVICE QDO2 OS1D04 PORTNAME BIGANG
LINK GIG2 QDIOETHERNET QDO2 ENABLEIPV6

HOME
125.0.0.34/16 GIG2
FEC0:0:0:1::/64 GIG2
```

5. In the following example, 504E is an OSA-Express Ethernet adapter that is configured to use the Queued Direct I/O Hardware Facility. With the VSWITCH VSWITCH1 keywords, the LQD4E link is

configured to allow IP address 125.1.7.20 to be used as management interface for VSWITCH. Network Management Systems can obtain bridge MIB information for VSWITCH1 using IP address 125.1.7.20.

```
DEVICE QD4E OSD 504E
LINK LQD4E QDIOETHERNET QD4E

HOME
125.1.7.20 LQD4E VSWITCH VSWITCH1
```

Usage Notes

- More than one home address can be associated with a link. The first home IPv4 address specified for a link is its primary one. However, if you are running MPRoute, only one home address can be associated with each link.
- **(IPv4 only)** If you are running a dynamic routing server (such as MPRoute), each link must have a unique IP address.
- **(IPv4 only)** The PRIMARYINTERFACE IP address is used as the source address in the IP header of an outgoing packet if no other source IP address can be found. If the PRIMARYINTERFACE statement is not specified, then the first address in the HOME list is the default local address, which is used for the GETHOSTID function.
- **(IPv4 only)** If a TCP application on the VM host system does not specify a local address value, the application receives the default local address, as described in the previous note. This value is set to the first address in the HOME list if a PRIMARYINTERFACE statement has not been associated with a link. For example, if the PRIMARYINTERFACE is set to LINK2, and the HOME list is as follows:

HOME	
9.0.28.3	LINK1
192.6.77.5	LINK2

then the default local address would be 192.6.77.5. If no PRIMARYINTERFACE is set, the default local address would be 9.0.28.3.

- When defining virtual IP addresses, observe the following rules and recommendations:
 - Code a primary virtual IP address first in the HOME list or on the PRIMARYINTERFACE statement to serve as the default local address.

In general, virtual IP addresses can be coded in any order in the HOME list; however, if you specify SOURCEVIPA on the ASSORTEDPARMS statement, the order of addresses is important with respect to how source IP addresses are used for outbound datagrams originating at the host. In this case, TCP/IP behaves as follows:

 - In the HOME list, the virtual IP address that most closely precedes a physical IP address is used as its source IP address.
 - If virtual IP addresses are coded after all physical IP addresses, no virtual addresses are used as source IP addresses.

Note: See the **Examples** section for more information about configuring the HOME statement when SOURCEVIPA is specified.

 - A virtual IP address must be a unique host address in the network and may not duplicate any physical IP address in the network.
 - More than one virtual IP address can be defined in one network or subnetwork.
 - You can use a virtual IP address as the primary (or only) destination for a z/VM server when the z/VM host is defined for your domain name server. A workstation on the network would use the z/VM server name (translated into the virtual IP address) to access applications on the z/VM server.
- While a virtual IP address can be assigned to each TCP/IP stack in one z/VM system, it is recommended that an internal point-to-point link (for example, virtual CTC) be defined between these stacks. This ensures the virtual IP address used for one z/VM TCP/IP stack that is attached to a failing device can be reached via another z/VM TCP/IP stack that is channel-attached to the same controller through another adapter, or to another controller across the point-to-point link.

- For more information about which routing protocol you can use to achieve non-disruptive TCP-connection fault tolerance, see [“Virtual IP Addressing \(VIPA\)”](#) on page 513.
- If you are using a nameserver to resolve host names via UDP and any of the related resolver configuration files have only one nameserver address coded that specifies a z/VM virtual IP address, the host the nameserver is running on must be configured to use SOURCEVIP.
- If you want a QDIOETHERNET or QDIOIP link to use IPv6 exclusively, do not use an IPv4 home address for that link.
- If an IPv6 home address is not specified for an IPv6-enabled link or if IPv6 home addresses are specified, but all addresses fail duplicate address detection, TCP/IP will enable autoconfiguration on the link using router advertisements.

If prefixes are configured for a link on the ROUTERADVPREFIX statement, then those prefixes will be used by TCP/IP to autoconfigure IPv6 home addresses. Otherwise, received router advertisements will be used to autoconfigure IPv6 home addresses.

- TCP/IP always autoconfigures a link-local address for each IPv6 enabled link.
- If a syntax error is found in a HOME statement, the remainder of the statement is ignored. Subsequent HOME statements in the same profile or OBEYFILE are processed.
- When a subnet mask is specified for IPv4 home addresses, the TCP/IP server automatically generates a direct static route to the subnet described by the IP address and mask. For IPv6 addresses, the TCP/IP server automatically generates a direct static route to the network described by the first 64 bits of the address. Unlike static routes added through the GATEWAY statement, these routes may be replaced by dynamic routing protocols if MPRoute is running.
- Providing management services for a virtual switch is independent of the virtual switch controller function, which is typically provided by TCP/IP stacks named DTCVSW1 or DTCVSW2.
- IPv6 virtual IP addresses cannot be auto-configured; the entire global IPv6 address must be specified on the HOME statement. The only prefix allowed with an IPv6 virtual IP address is /128.

More Information

- [“ASSORTEDPARMS Statement”](#) on page 529
- [“Attaching network devices to the TCPIP virtual machine”](#) on page 519
- [“PRIMARYINTERFACE Statement”](#) on page 606

ICMPERRORLIMIT Statement

Use the ICMPERRORLIMIT statement to specify the maximum number of ICMPv6 error messages that can be sent on a link per second.



Operands

msgs_per_second

Specifies the number of ICMPv6 error messages that may be sent on a link per second. The valid range is 1-50 messages/second. The default is 10. An internal algorithm is used to allow bursts of ICMP errors while limiting the long term rate.

INFORM Statement

Use the INFORM statement to define a list of users (called the INFORM list) who are to be sent messages in case of serious run-time conditions.



Operands

user_id

Defines a user to be informed in case of serious run-time error conditions.

Examples

The following example shows how to inform users OPERATOR, TCPMAINT, and MAINT of serious errors.

```
Inform
  OPERATOR TCPMAINT MAINT
EndInform
```

Usage Notes

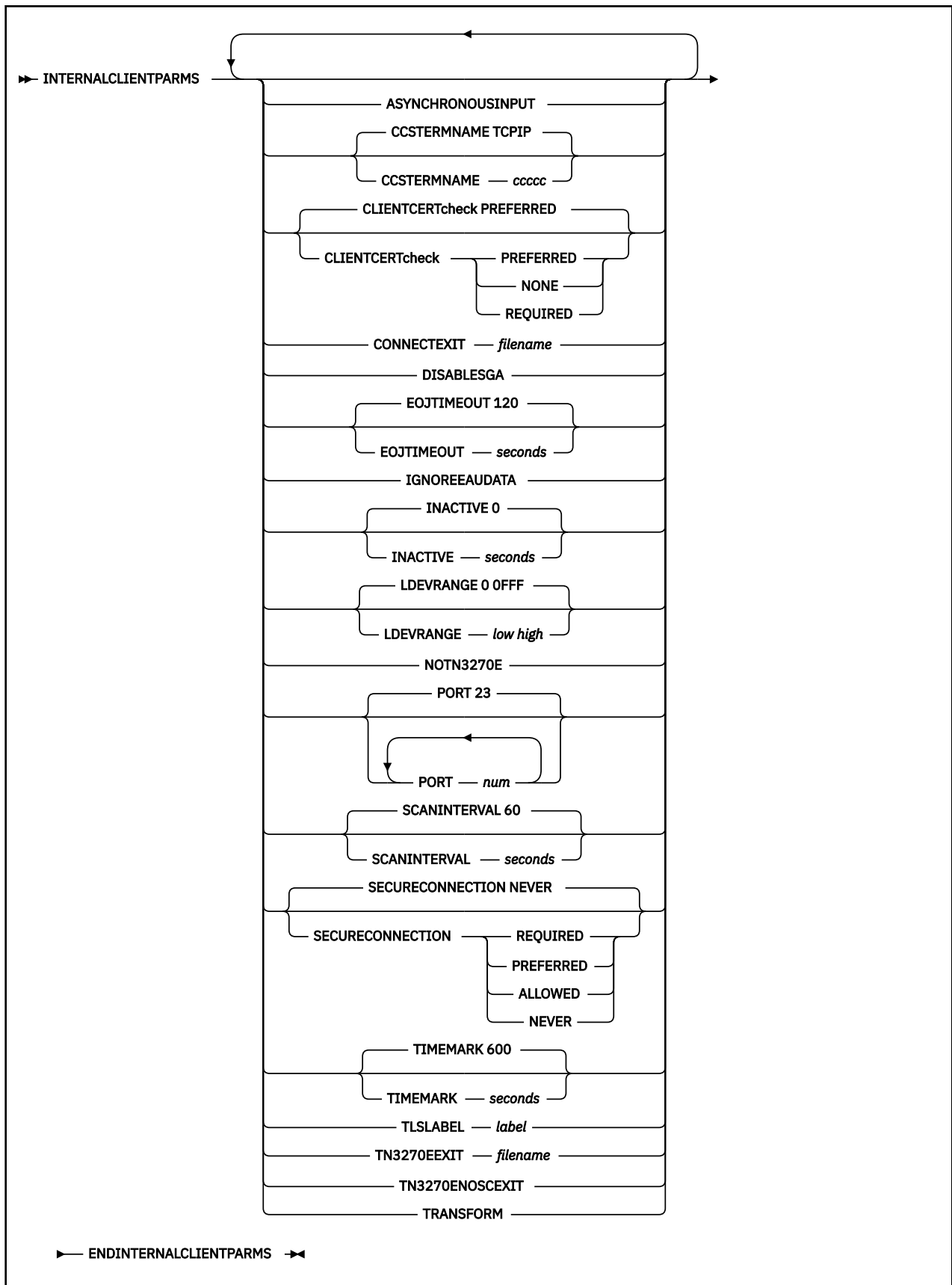
- The ENDINFORM statement ends the INFORM statement. If ENDINFORM is omitted, all subsequent statements will be interpreted as user IDs.
- When a serious error is detected, each user in the current INFORM list is sent a short descriptive message. For example, when the number of objects in any pool displayed by the NETSTAT POOLSIZE command falls below its permit size, each user in the list is notified.
- The first INFORM statement of each configuration file replaces the existing INFORM list. Subsequent INFORM statements in the same file add users to the list.
- If no valid user IDs are specified, the statement is ignored.
- Specifying INFORM immediately followed by ENDINFORM will clear the INFORM list.

More Information

- [“Free Pool Statements” on page 521](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

INTERNALCLIENTPARMS Statement

The INTERNALCLIENTPARMS statement is used to configure the Telnet server, an internal client of the TCPIP virtual machine. A maximum of 500 characters can be used to specify the INTERNALCLIENTPARMS.



Operands

ASYNCHRONOUSINPUT

ASYNCHINPUT

ASYNCSINPUT

For Telnet LINEMODE connections, causes the Telnet server to signal an attention interrupt to the associated virtual machine, when input is received from the client and the virtual machine has not issued a read. This usually causes the virtual machine to issue a read, allowing the user input to be presented. If this option is not specified, the Telnet server holds client input until the associated virtual machine issues a read.

CCSTERMNAME *cccc*

String 1 to 5 characters in length specifying the terminal name prefix for line-mode Telnet sessions.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

CONNECTEXIT *filename*

The name of the Telnet session connection exit to be loaded. The exit will be called every time a Telnet session connection is established unless the TN3270ENOSCEXIT parameter has been supplied via the OBEYFILE command or as a parameter on the INTERNALCLIENTPARMS statement in the TCP/IP configuration file.

The CONNECTEXIT can be used to control system access based on the client's IP address and the target port number. The exit can also be used to specify an initial CP command (such as DIAL VTAM) to be simulated for a client with a transparent mode session.

The search sequence is:

1. The GLOBAL LOADLIB list,
2. *filename* TEXT on any accessed disk,
3. The GLOBAL TXTLIB list.

The CONNECTEXIT parameter is ignored if it is supplied via the OBEYFILE command. The exit interface is described in the *z/VM: TCP/IP Programmer's Reference*.

Sample copies of the Telnet session connection exit files (SCEXIT EXEC, SCEXIT ASSEMBLE and SCEXIT TEXT) are supplied as softcopy files (SCEXIT SAMPEXEC, SCEXIT SAMPASM, and SCEXIT TEXTSAMP, respectively) on the TCPMAINT 591 minidisk. Consult the *z/VM: TCP/IP Programmer's Reference* for details about SCEXIT parameter list and parameter descriptions.

DISABLESGA

Suppresses the transmission of GO AHEADS by Telnet, which is negotiated by both client and server. Using DISABLESGA reduces the overhead for a full duplex terminal and a full duplex connection.

EOJTIMEOUT *seconds*

Sets the EOJTIMEOUT interval. This parameter is used in conjunction with TN3270E printer support, and causes an EOJ header to be sent to a printer if no such header is sent within the specified number of seconds.

Specify *seconds* as a positive integer in the range of 1 through 99,999,999. If *seconds* is not within the range of accepted values or this operand is omitted, the default of 120 seconds (two minutes) is used.

IGNOREEAUDATA

Causes the Telnet server to ignore any data associated with Erase All Unprotected (EAU) commands in the data stream received from the host. Ordinarily, any such data is forwarded to the client. Some Telnet clients enforce the restriction that there can be no data associated with an EAU command and require this option in order to function properly.

INACTIVE *seconds*

Defines an interval (in seconds) after which the Telnet server closes a connection due to inactivity. In the context of this operand, a connection is considered to be inactive if no data is transmitted over that connection for the specified amount of time.

Specify *seconds* as zero (0) to signify no inactivity interval is to be in effect (that is, connections are not to be closed due to inactivity), or as a positive integer in the range of 1 through 99,999,999. If *seconds* is not within the range of accepted values or this operand is omitted, the default of zero (0) is used.

Note: Telnet protocol commands transmitted over a connection do not affect inactivity timing.

LDEV RANGE *low high*

Hexadecimal logical device number range between 0 and FFFF to be used for incoming Telnet connections. Do not set the end of the range larger than the maximum logical device number defined by the CP SET MAXLDEV command. Since logical device numbers are unique within the VM system, there is no guarantee that other service machines will not use the same device range that is assigned to TCP/IP.

If LDEV RANGE is not specified, logical device numbers in the range from 0 to 0FFF will be used.

NOTN3270E

Prevents the Telnet server from negotiating sessions based on the TN3270E protocol. Some Telnet clients might not handle TN3270E negotiation correctly, in which case this parameter can be used to allow them to function correctly. However, Telnet-based printer sessions are not supported if this parameter is specified.

PORT *num*

Accepts incoming Telnet requests on a specified port number rather than the default port 23. This parameter may be specified multiple times to accept incoming Telnet requests on any of several different ports. The port numbers specified should have corresponding PORT statements that reserve them for the special user identifier INTCLIEN, which represents the Telnet server.

The PORT parameter is ignored if it is supplied via the OBEYFILE command.

SCANINTERVAL *seconds*

Defines the interval at which the Telnet server checks connections to determine whether a TIMEMARK should be sent over a connection, or if a connection should be closed due to an elapsed interval of inactivity.

Specify *seconds* as a positive integer in the range of 1 through 60. If *seconds* is not within the range of accepted values or this operand is omitted, the default of 60 seconds (one minute) is used.

To facilitate appropriate timing-based actions, the SCANINTERVAL value is adjusted by the Telnet server to match the smallest interval established among the following:

- EOJTIMEOUT
- INACTIVE
- TIMEMARK
- SCANINTERVAL
- 60 seconds (a constant)

SECURECONNECTION REQUIRED

All Telnet connections must be secured either statically or dynamically.

SECURECONNECTION PREFERRED

If a Telnet connection is not secured statically, the Telnet server will initiate the request for TLS to be used; if the client is unable to use TLS, the connection proceeds as a clear connection. Statically secured connections are not affected by this option.

SECURECONNECTION ALLOWED

If a Telnet connection is not secured statically, the Telnet server will use TLS only when the request to use TLS is initiated by the client. Statically secured connections are not affected by this option.

SECURECONNECTION NEVER

All Telnet connections must be clear connections.

TIMEMARK *seconds*

Defines an interval (in seconds) after which the Telnet server is to send a TIMEMARK option on a given connection. TIMEMARK options issued in this context serve to verify the client associated with a probed connection is operational.

Specify *seconds* as zero (0) to signify no TIMEMARKs are to be sent by the Telnet server, or as a positive integer in the range of 1 through 99,999,999. If *seconds* is not within the range of accepted values or this operand is omitted, the default of 600 seconds (10 minutes) is used.

When TIMEMARKs are in use, the Telnet server sends a TIMEMARK option over a connection once SCANINTERVAL processing has determined the time elapsed (since the last activity for the connection) is greater than the defined TIMEMARK interval. The receipt of a client TIMEMARK response confirms the connection should be maintained, and the response is otherwise ignored. If a TIMEMARK response is not received on a connection and a subsequent TIMEMARK interval then passes with no additional activity, the connection is closed by the Telnet server, under the assumption the client host can no longer respond (due to networking or other problems).

Note:

1. The TIMEMARK value established can affect that used for SCANINTERVAL processing.
2. Connections that are responsive to TIMEMARK probes may still be closed due to inactivity, as directed by the INACTIVE timing operand.

TLSLABEL *label*

Specifies the TLS label to be used by the Telnet server when securing connections using TLS.

Note: The TLS label can be no more than 8 characters, and must be comprised of only uppercase, alphanumeric characters.

TN3270EXIT *filename*

The name of the Telnet printer management exit. The exit is called every time a Telnet printer session is established or terminated.

The TN3270EXIT can be used to control system access based on the client's IP address and port number, the local port number, the logical unit name associated with the session by the client, and the user identifier and virtual device address associated with the session through the TN3270E statement (see "[TN3270E Statement](#)" on page 622).

The search sequence is:

1. The GLOBAL LOADLIB list,
2. *filename* TEXT on any accessed disk,
3. The GLOBAL TXTLIB list.

The TN3270EXIT parameter is ignored if it is supplied via the OBEYFILE command. The exit interface is described in the [z/VM: TCP/IP Programmer's Reference](#).

Sample copies of the Telnet printer management exit files (PMEXIT EXEC, PMEXIT ASSEMBLE and PMEXIT TEXT) are supplied as softcopy files (PMEXIT SAMPEXEC, PMEXIT SAMPASM, and PMEXIT TEXTSAMP, respectively) on the TCPMAINT 591 minidisk. Consult the [z/VM: TCP/IP Programmer's Reference](#) for details about PMEXIT parameter list and parameter descriptions.

TN3270ENOSCEXIT

Prevents the Telnet server from calling the session connection exit for telnet printer sessions only. Otherwise, the session connection exit will be called, if it is loaded, for all telnet sessions.

TRANSFORM

Causes the Telnet server to load a 3270 transform program. File TNSIMHPI TEXT must be accessible by the server, and additional virtual storage might be needed. This file is available only with third-party products; it is not supplied with TCP/IP for VM. The TRANSFORM parameter is ignored if it is supplied via the OBEYFILE command.

Examples

The following example shows that TNEXIT1 TEXT is to be loaded and used as the Telnet session connection exit.

```
InternalClientParms
  ConnectExit TNEXIT1
EndInternalClientParms
```

Usage Notes

- If a parameter name is misspelled or if the value specified is not valid, the parameter is ignored and the default is used.
- When using the OBEYFILE command to modify the INTERNALCLIENTPARMS statement, keep these rules in mind:
 - The values of all parameters except CONNECTEXIT, PORT, TN3270EEXIT, and TRANSFORM may be changed.
 - A change to any read-only disk will cause the CONNECTEXIT and TN3270EEXIT interface to be reloaded during Obeyfile processing for INTERNALCLIENTPARMS.
 - An INTERNALCLIENTPARMS parameter that may be changed but is not specified assumes its default value or setting.
- Communication with SNA/CCS terminals must be enabled in order for the Telnet server to support line-mode Telnet sessions. This can be accomplished by adding the CP ENABLE SNA command to a server profile exit or the global profile exit, TCPRUNXT. For more information, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.
- To disable line-mode Telnet support, ensure that the *CCS operand is not specified for any IUCV statements that are included in the TCP/IP server CP directory entry. Also, ensure that the CP ENABLE SNA command is not employed as previously described.
- Prior to customizing the telnet exits described in this section (CONNECTEXIT and TN3270EXIT), ensure that you have reviewed the exit limitations and customization recommendations presented in [“Customizing Server-specific Exits”](#) on page 49.
- The CLIENTCERTCHECK, SECURECONNECTION, and TLSLABEL options can be modified by an INTERNALCLIENTPARMS statement using the NETSTAT OBEY or OBEYFILE command.
- When the SECURECONNECTION operand is omitted or is set to NEVER, the CLIENTCERTCHECK operand is not valid. The CLIENTCERTCHECK value will be set to NONE (the default).

More Information

- [“PORT Statement”](#) on page 603
- [“TN3270E Statement”](#) on page 622

IPROUTEPOOLSIZE Statement

Use the IPROUTEPOOLSIZE statement to set the initial number of IPv4 IP route control blocks and the initial number of IPv6 IP route control blocks. IP route control blocks are used to hold information about IP routes.



Operands

number

The initial number of IP route control blocks in the free pool.

Examples

The following example shows an IPROUTEPOOLSIZE statement that defines the number of IP route control block to be the default of 600.

```
IProutePoolSize    600
```

Usage Notes

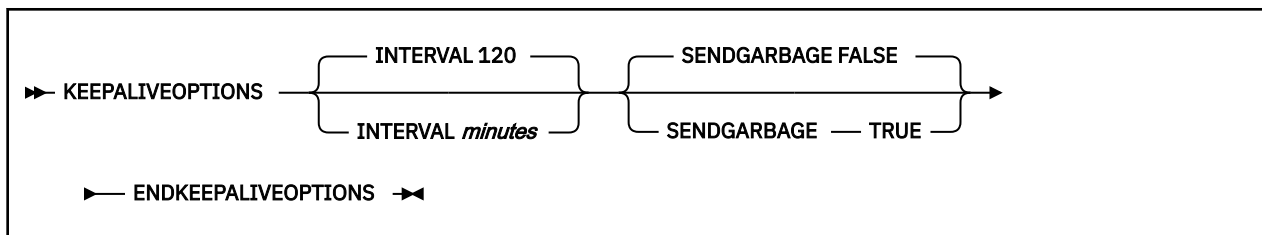
- Each entry in the IP routing table, whether created using the GATEWAY statement, by Internet Control Message Protocol (ICMP) Redirect, or by MPRoute, requires one IP route control block. When an entry is deleted from the IP routing table, the corresponding IP route control block is not immediately returned to the pool, if the IP-down route cache or a TCP connection refers to it. The control block is returned to the pool the next time TCP/IP tries to use the saved reference and finds that the entry is no longer valid.
- The system will attempt to dynamically allocate 10% more IP route control blocks any time the IP route control block free pool becomes empty. You can use the NETSTAT POOLSIZE command to monitor how many IP route control blocks your system is using. To avoid dynamic allocation of IP route control blocks during operation, use the IPROUTEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- “GATEWAY Statement” on page 568
- z/VM: TCP/IP User's Guide* for the NETSTAT command

KEEPALIVEOPTIONS Statement

Use the KEEPALIVEOPTIONS statement to specify the operating parameters of the TCP/IP keep-alive mechanism. The parameters apply to all TCP connections for which keep-alive has been activated, through either the setsockopt() call of the C socket interface or the TcpOption call of the Pascal interface.



Operands

INTERVAL *minutes*

The number of minutes TCP/IP waits after last receiving a packet for a TCP connection before it sends a keep-alive packet for it. The default is 120 minutes (2 hours).

SENDGARBAGE

Specifies whether the keep-alive packets sent by TCP/IP contain one byte of random data.

FALSE

The keep-alive packet will not contain random data. This is the default.

TRUE

The keep-alive packet will contain one byte of random data and an invalid sequence number, assuring that the data is not accepted by the remote TCP/IP.

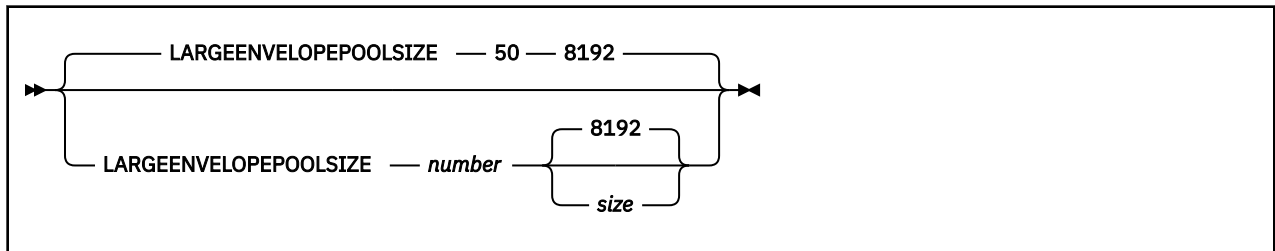
Usage Notes

- Some hosts cannot properly respond to keep-alive packets containing no data. If your network includes such hosts, set the SENDGARBAGE parameter to TRUE.
- The ENDKEEPAVIVEOPTIONS statement specifies the end of the KEEPAVIVEOPTIONS information. If it is omitted, subsequent entries will generate error messages.

LARGEENVELOPEPOOLSIZE Statement

Use the LARGEENVELOPEPOOLSIZE statement to set the initial number and size of large envelopes. Large envelopes are used to hold UDP datagrams larger than 2,048 bytes while they are being sent and while they are waiting for an application program to receive them. They are also used to hold IP datagram fragments during reassembly.

A large envelope is used only if a packet does not fit into a small envelope.



Operands

number

The initial number of large envelopes in the free pool. The default is 50.

size

The size of each large envelope (in bytes); this *size* determines the maximum number of bytes that can be held by a large envelope. The default is 8192. Only the following values or their alternatives (shown in parenthesis) may be specified:

512
 1024 (1K)
 2048 (2K)
 4096 (4K)
 8192 (8K)
 16384 (16K)
 32768 (32K)
 65535 (64K)

Examples

The following example shows a LARGEENVELOPEPOOLSIZE statement that defines the default number and size of large envelopes.

```
LargeEnvelopePoolSize 50 8192
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- A large envelope size of 8192, which is the default, actually accommodates packets up to 9,216 bytes long.
- If any *max_packet_size* on the GATEWAY statement is greater than 2,048, then additional large envelopes will be needed for applications such as FTP.
- If *size* is not specified, large envelopes can hold up to 9,216 bytes of data, so packets up to 9,216 bytes can be sent and received.
- Specify the *size* operand to establish the size of the largest packet that can be sent and received.

Note: The specified *size* value must equal or exceed that specified for regular-sized envelopes (defined by the ENVELOPEPOOLSIZE statement). Failure to do so will result in TCP/IP stack initialization errors.

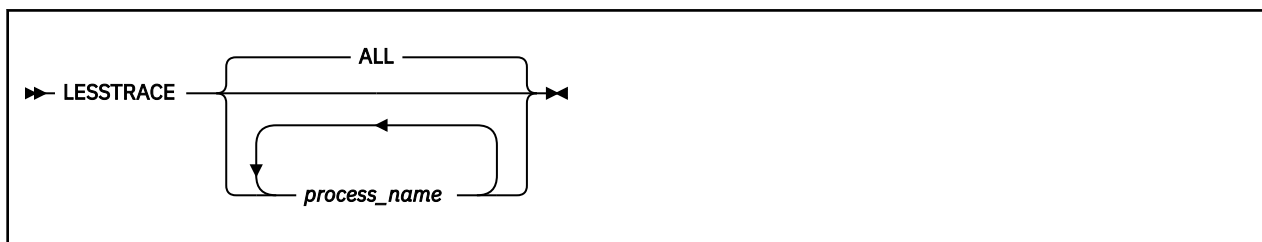
- Matching *size* values should be specified for each of the two hosts for which a CTC connection is defined.
- The system will attempt to dynamically allocate 10% more large envelopes any time the large envelope free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many large envelopes your system is using. To avoid dynamic allocation of large envelopes during operation, use the LARGEENVELOPEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [“DATABUFFERPOOLSIZE Statement” on page 537](#)
- [“ENVELOPEPOOLSIZE Statement” on page 564](#)
- [“GATEWAY Statement” on page 568](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

LESSTRACE Statement

Use the LESSTRACE statement to turn off detailed run-time tracing for the specified internal TCP/IP processes.



Operands

ALL

Suppresses detailed run-time tracing information for all processes.

process_name

Suppresses detailed run-time tracing information for the named process. For valid process names, refer to [Table 46 on page 624](#).

Usage Notes

- If no process names are specified, detailed tracing is suppressed for all processes.
- LESSTRACE affects only detailed tracing. To turn off basic tracing, use NOTRACE.

More Information

- [“TRACE Statement” on page 623](#)
- [“MORETRACE Statement” on page 595](#)
- [“NOTRACE Statement” on page 597](#)
- [“TRACEONLY Statement” on page 625](#)

MAXRESTART Statement

The MAXRESTART statement specifies the maximum number of times the TCP/IP server will attempt to restart a user after the first autolog attempt.

**Operands*****count***

Specifies the maximum number of times the TCP/IP server will attempt to restart a user after the first autolog attempt. This is an integer between 0 and 2,147,483,647. The default is 5.

Examples

The following example shows a MAXRESTART statement that specifies 5 as the maximum number of times the TCP/IP server will attempt to restart a user after the first autolog attempt.

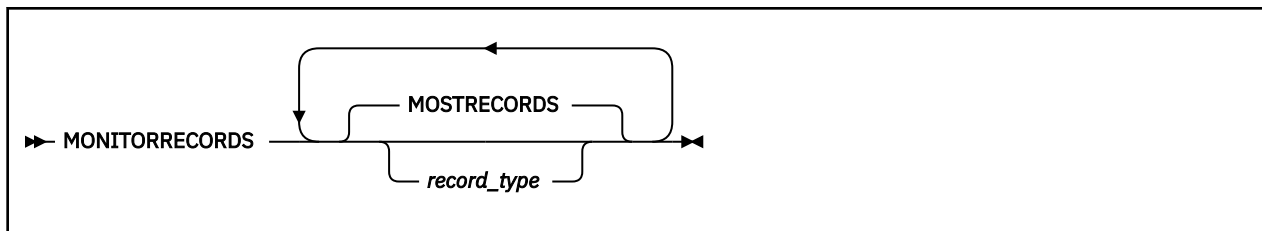
```
MAXRESTART 5
```

Usage Notes

- If a user is restarted MAXRESTART times, the TCP/IP server will send a message to the INFORM list and stop restarting the user. After that, you can tell the TCP/IP server to attempt to restart the user another MAXRESTART times by dynamically specifying the user's PORT statement without the NOAUTOLOG operand.
- If a user is restarted less than MAXRESTART times and then runs successfully for 24 hours, the TCP/IP server will reset the user's restart count to zero and begin counting again with the next restart.
- The first autolog attempt by the TCP/IP server after the TCP/IP server is initialized or after the TCP/IP server processes a user's PORT statement without the NOAUTOLOG operand does not count towards the user's MAXRESTART value.

MONITORRECORDS Statement

Use the MONITORRECORDS statement to select the monitor data records that are produced by TCPIP. This statement is ignored if it is supplied through the OBEYFILE command interface.



Operands

record_type

Specifies the name of a monitor data record type and is one of the following:

Type

Records produced

ALLRECORDS

all

CPU

CPU consumption

CLIENTS

client activity

HOMES

home address configuration

HOMESIPV6

IPv6 home address configuration

LINKS

link configuration and activity

MIB

management information base data

MOSTRECORDS

all except Scheduler (default)

POOLS

storage pool configuration and use

SCHEDULE

Scheduler activity

TCP-SESSIONS

TCP session activity

TREES

hash tree size

UDP-SESSIONS

UDP session activity

Usage Notes

- The MONITORRECORDS statement determines which performance monitor data records are produced, if any. The list of record names is terminated by end of input or by another configuration statement. An empty list, that is, just the MONITORRECORDS statement alone, enables production of the monitor records in the MOSTRECORDS record type alias.
- In order for this statement to take effect, the TCPIP virtual machine must be authorized to create monitor data records by having an OPTION APPLMON statement in its User Directory entry.
- To collect monitor data, the APPLDATA class must be enabled for both SAMPLE and EVENT recording for the TCPIP virtual machine and a monitor writer must be active. For example, these CP commands would cause monitor data produced by the virtual machine named TCPIP to be collected.

```
MONITOR SAMPLE ENABLE APPLDATA USER TCPIP
MONITOR EVENT  ENABLE APPLDATA USER TCPIP
```

These CP commands would cause any virtual machine's monitor data to be collected.

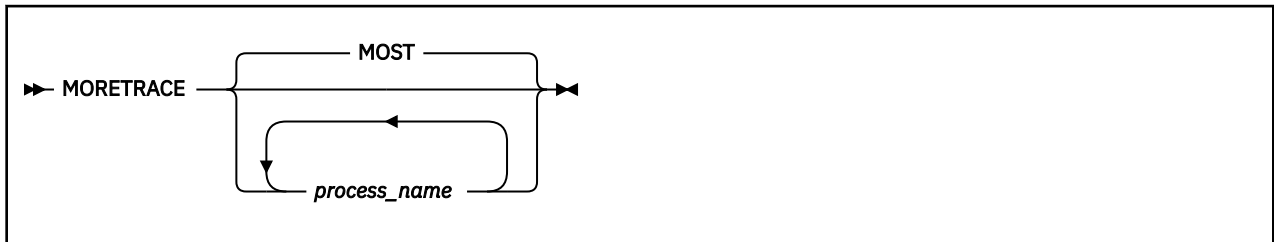
```
MONITOR SAMPLE ENABLE APPLDATA ALL
MONITOR EVENT  ENABLE APPLDATA ALL
```

- To ensure that TCPIP configuration information is included in the monitor data file, start the monitor and a monitor writer before initializing the TCPIP virtual machine.

For more information about performance monitoring, see [z/VM: Performance](#).

MORETRACE Statement

Use the MORETRACE statement to turn on detailed run-time tracing for the specified internal TCP/IP processes.



Operands

MOST

Enables detailed run-time tracing information for all processes except INITIALIZE, SCHEDULER, and TIMER.

process_name

Enables detailed run-time tracing information for the named process. For valid process names, refer to [Table 46 on page 624](#).

Usage Notes

- If no process name or an invalid process name is specified, detailed tracing is enabled for the processes in the MOST process name alias.
- If a TRACEONLY statement is used, the trace is restricted to the processes that are related to the selected users, devices, or IP addresses.

More Information

- [“TRACE Statement” on page 623](#)
- [“TRACEONLY Statement” on page 625](#)
- [“LESSTRACE Statement” on page 592](#)
- [“NOTRACE Statement” on page 597](#)

NCBPOOLSIZE Statement

Use the NCBPOOLSIZE statement to set the initial number of IPv6 neighbor control blocks. Neighbor control blocks are used to hold information about neighboring nodes. For example, the neighbor cache holds information about the relationship between IPv6 addresses and network addresses. Each neighbor control block requires 68 bytes.



Operands

number

The initial number of neighbor control blocks in the free pool.

Examples

The following example shows an NCBPOOLSIZE statement that defines the number of neighbor control blocks to be the default of 1500.

```
NcbPoolSize    1500
```

Usage Notes

- The system will attempt to dynamically allocate 10% more neighbor control blocks any time the neighbor control block free pool becomes empty. You can use the NETSTAT POOLSIZE command to monitor how many neighbor control blocks your system is using. To avoid dynamic allocation of neighbor control blocks during operation, use the NCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

For the NETSTAT command, [z/VM: TCP/IP User's Guide](#).

NOSCREEN Statement

Use the NOSCREEN statement to divert run-time trace output from the TCPIP virtual machine console back to the most recently used TRACE file.

```
➤ NOSCREEN ➤
```

Operands

The NOSCREEN statement has no operands.

Usage Notes

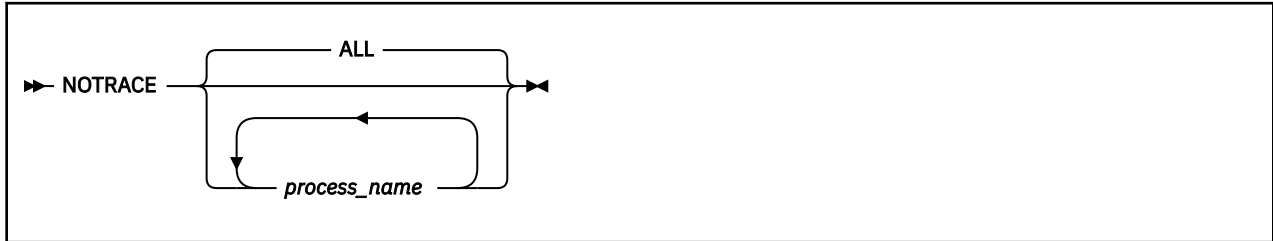
The NOSCREEN statement diverts trace information currently directed to the TCPIP console back to the most recently used trace file; any data previously contained in the trace file is destroyed. If no trace file was previously identified via a FILE statement, information will be written to the file **DEBUG TRACE A**.

More Information

- [“TRACE Statement” on page 623](#)
- [“FILE Statement” on page 565](#)

NOTRACE Statement

Use the NOTRACE statement to turn off basic and detailed run-time tracing for the specified TCP/IP processes.



Operands

ALL

Suppresses basic and detailed run-time tracing information for all processes.

process_name

Suppresses basic and detailed run-time tracing information for the named process. For valid process names, refer to [Table 46 on page 624](#).

Examples

The following example shows a NOTRACE statement that turns off all tracing for the Telnet process.

```
NOTRACE TELNET
```

Usage Notes

- If no process names are specified, basic and detailed tracing is suppressed for all processes.
- Processing of the NOTRACE statement will not cause an open trace file to be closed; a FILE or SCREEN statement must be used to close an open file.

More Information

- [“TRACE Statement” on page 623](#)
- [“LESSTRACE Statement” on page 592](#)
- [“MORETRACE Statement” on page 595](#)

OBEY Statement

Use the OBEY statement to define which users can use the following privileged TCP/IP functions:

- The OBEYFILE command
- The NETSTAT BLOCK command
- The NETSTAT CP command
- The NETSTAT DELARP command
- The NETSTAT DELNEIGHBOR command
- The NETSTAT DROP command
- The NETSTAT OBEY command
- The NETSTAT RESETDOS command
- The NETSTAT RESETPOOL command
- The NETSTAT UNBLOCK command

PACKETTRACESIZE

- Any program that uses rawIP functions
- MSG functions of the FTP and MPRoute servers

The first OBEY statement of each configuration file replaces the existing OBEY list with the new list. Subsequent OBEY statements in the same file add users to the list.



Operands

user_id
A user who is authorized to use privileged TCP/IP functions.

Examples

The following example shows an OBEY statement that identifies eight users who are to be obeyed.

```
Obey
  OPERATOR TCPMAINT SNMPD SNMPQE MPRUTE REXECD
EndObey
```

Usage Notes

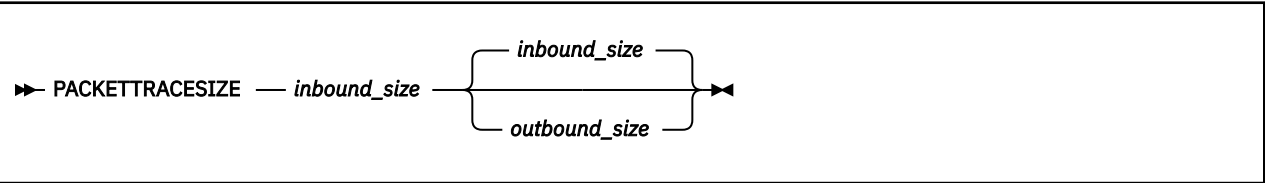
- The ENDOBEY statement ends the OBEY statement. If ENDOBEY is omitted, all subsequent statements are considered to be entries in the OBEY list.
- TCP/IP servers that must be in the OBEY list in order to function properly are:
 - MPRUTE
 - REXECD
 - SNMPD
 - SNMPQE
- If no valid user IDs are specified, the statement is ignored.
- Specifying OBEY immediately followed by ENDOBEY will clear the OBEY list.

More Information

- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

PACKETTRACESIZE Statement

Use the PACKETTRACESIZE statement to specify the amount of data from inbound and outbound packets that is to be included in TRSOURCE packet traces.



Operands

inbound_size

The amount of data from each inbound packet to be collected. The size may not exceed 32724. Specify a value of 0 to turn off inbound packet tracing.

outbound_size

The amount of data from each outbound packet to be collected. The size may not exceed 32724. Specify a value of 0 to turn off outbound packet tracing. If this operand is not specified, the *inbound_size* value is used.

Examples

To collect packet trace data sufficient to understand the flow of TCP packets through a TCP/IP virtual machine called TCP/IP for an Ethernet adapter named ETH0, do the following:

1. Issue the following commands to enable the trace:

```
NETSTAT OBEY PACKETTRACESIZE 64
NETSTAT OBEY TRACEONLY ETH0 ENDTRACEONLY
```

2. Issue the following Control Program commands to start data collection:

```
TRSOURCE ID TCP TYPE GT BLOCK FOR USER TCP/IP
TRSOURCE ENABLE ID TCP
```

3. Invoke the function you want to trace. When the function has completed, issue the following commands to stop the TCP/IP trace:

```
NETSTAT OBEY PACKETTRACESIZE 0
NETSTAT OBEY TRACEONLY ENDTRACEONLY
```

4. Issue the following Control Program command:

```
TRSOURCE DISABLE ID TCP
```

5. Analyze the resulting TRF file. You can use the TRACERED command to display the trace data.

Example: If the TRF spool file identifier is 1234, issue the following command:

```
TRACERED 1234 CMS TCP TRACE A ( ALL
```

Usage Notes

- In addition to setting a non-zero value for PACKETTRACESIZE, you must use the TRACEONLY statement to enable selective tracing for the device or devices whose packets you want to trace.

Note: The TRSOURCE method of packet tracing requires a device name on the TRACEONLY statement. Specifying an IP address on the TRACEONLY statement will not result in TRSOURCE data being collected for that IP address.

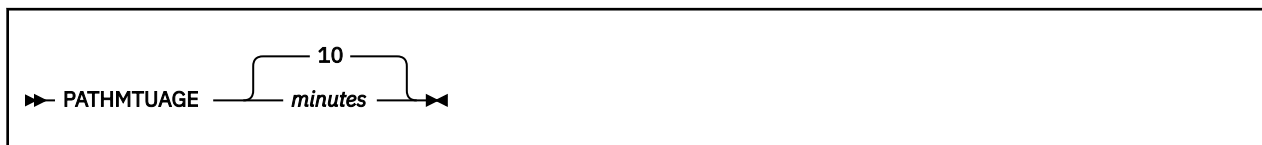
- To collect the packet trace data, use the Control Program TRSOURCE command to enable TYPE GT BLOCK tracing for the TCP/IP virtual machine.
- To completely disable packet tracing, set PACKETTRACESIZE to 0, terminate selective tracing of all devices, and issue a TRSOURCE DISABLE command to terminate type GT TRSOURCE data collection.

More Information

- [z/VM: CP Commands and Utilities Reference](#) for the TRSOURCE and TRACERED commands
- “TRACEONLY Statement” on page 625
- [z/VM: TCP/IP Diagnosis Guide](#) for the IPFORMAT command.

PATHMTUAGE Statement

Use the PATHMTUAGE statement to specify how long path MTU discovery information is retained. After the specified time has expired, since the start of path MTU discovery for a given destination, the path MTU discovery information will be disposed. The next time traffic is sent to the destination, path MTU discovery is restarted to determine changes to the path MTU size.



Operands

minutes

The number of minutes path MTU discovery information will be retained for all routes. Specify *minutes* as an integer between 5 and 43200 (30 days). The default is 10 minutes.

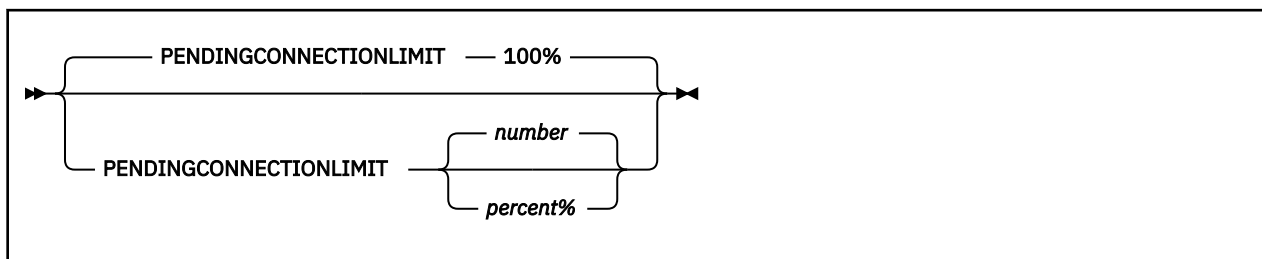
More Information

- “ASSORTEDPARMS Statement” on page 529
- “DEVICE and LINK Statements” on page 539
- “IFCONFIG Command” on page 631
- NETSTAT command, see *z/VM: TCP/IP User's Guide*

PENDINGCONNECTIONLIMIT Statement

Use the PENDINGCONNECTIONLIMIT statement to define the maximum number of half-open connections that are allowed at any given time. When a new half-open connection causes this limit to be exceeded, a random current half-open connection is dropped and a SynFlood denial-of-service attack is declared.

Note: When the limit is dynamically changed to a value that is lower than the previous value and there are currently more connections in the list than allowed by the new limit, the oldest connections will be dropped until the new limit is reached.



Operands

number

The maximum number of half-open connections allowed at any given time.

percent%

The maximum number of half-open connections allowed at any given time, specified as a percentage of the current TCBPOOLSIZE. The default is 100%.

Examples

1. The following example shows a PENDINGCONNECTIONLIMIT statement specifying the maximum number of half-open connections to be 256.

```
PENDINGCONNECTIONLIMIT 256
```

2. The following example shows a PENDINGCONNECTIONLIMIT statement specifying the maximum number of half-open connections to be 10% of the current TCBPOOLSIZE.

```
PENDINGCONNECTIONLIMIT 10%
```

Usage Notes

- The PENDINGCONNECTIONLIMIT value does not change when TCBPOOLSIZE changes. The PENDINGCONNECTIONLIMIT value can be changed only through the OBEYFILE command.
- A PENDINGCONNECTIONLIMIT value of 0 indicates there is no limit on the number of half-open connections allowed at any given time.
- Use the NETSTAT DOS command to display the current limit of half-open connections.

More Information

- “OBEYFILE Command” on page 644
- *z/VM: TCP/IP User's Guide* for the NETSTAT DOS and NETSTAT RESETDOS commands

PERMIT Statement

Use the PERMIT statement to identify users who are allowed to use TCP/IP services.

The first PERMIT statement of each configuration file replaces the existing PERMIT list with the new list. Subsequent PERMIT statements in the same file add users to the list.



Operands

user_id

A user who is allowed to use general TCP/IP services.

Usage Notes

- The PERMIT statement is effective only when PERMITTEDUSERONLY is specified on the ASSORTEDPARMS statement.
- The ENDPERMIT statement ends the PERMIT statement. If ENDPERMIT is omitted, all subsequent statements will be interpreted as user IDs.
- A user who is specified on the AUTOLOG, INFORM, OBEY, or PORT statements does not need to be specified on the PERMIT statement.
- If a user is specified on both the PERMIT and RESTRICT statements, the user will not be able to use TCP/IP services.
- If an unauthorized user attempts to use TCP/IP services, the message

```
date time Unauthorized TCP/IP access attempt by userid
```

is logged by the TCPIP virtual machine in the stack console or in the trace file (whichever is active) and to a special authorization audit file named TCPIP AUTHLOG A.

You can override the audit file specification by including an appropriate :Authlog. tag entry in a customized DTCPARMS file definition for the TCP/IP server.

- If no valid user IDs are specified, the statement is ignored.
- Specifying PERMIT immediately followed by ENDPERMIT will clear the PERMIT list.

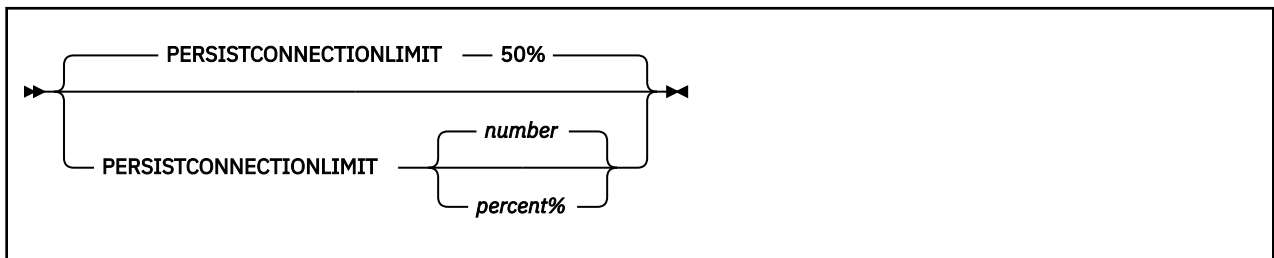
More Information

- [“ASSORTEDPARMS Statement” on page 529](#)
- [“RESTRICT Statement” on page 608](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)

PERSISTCONNECTIONLIMIT Statement

Use the PERSISTCONNECTIONLIMIT statement to define the maximum number of connections in TCP persist state at any given time. When a new connection in persist state causes this limit to be exceeded, the oldest current connection in persist state is dropped and a ZeroWin denial-of-service attack is declared.

Note: When the limit is dynamically changed to a value that is lower than the previous value and there are currently more connections in the list than allowed by the new limit, the oldest connections will be dropped until the new limit is reached.



Operands

number

The maximum number of connections in persist state allowed at any given time.

percent%

The maximum number of connections in persist state allowed at any given time, specified as a percentage of the current TCBPOOLSIZE. The default is 50%.

Examples

1. The following example shows a PERSISTCONNECTIONLIMIT statement specifying the maximum number of connections in persist state to be 256.

```
PERSISTCONNECTIONLIMIT 256
```

2. The following example shows a PERSISTCONNECTIONLIMIT statement specifying the maximum number of connections in persist state to be 50% of the current TCBPOOLSIZE.

```
PERSISTCONNECTIONLIMIT 50%
```

Usage Notes

- The PERSISTCONNECTIONLIMIT value does not change when TCBPOOLSIZE changes. The PERSISTCONNECTIONLIMIT value can be changed only through the OBEYFILE command.

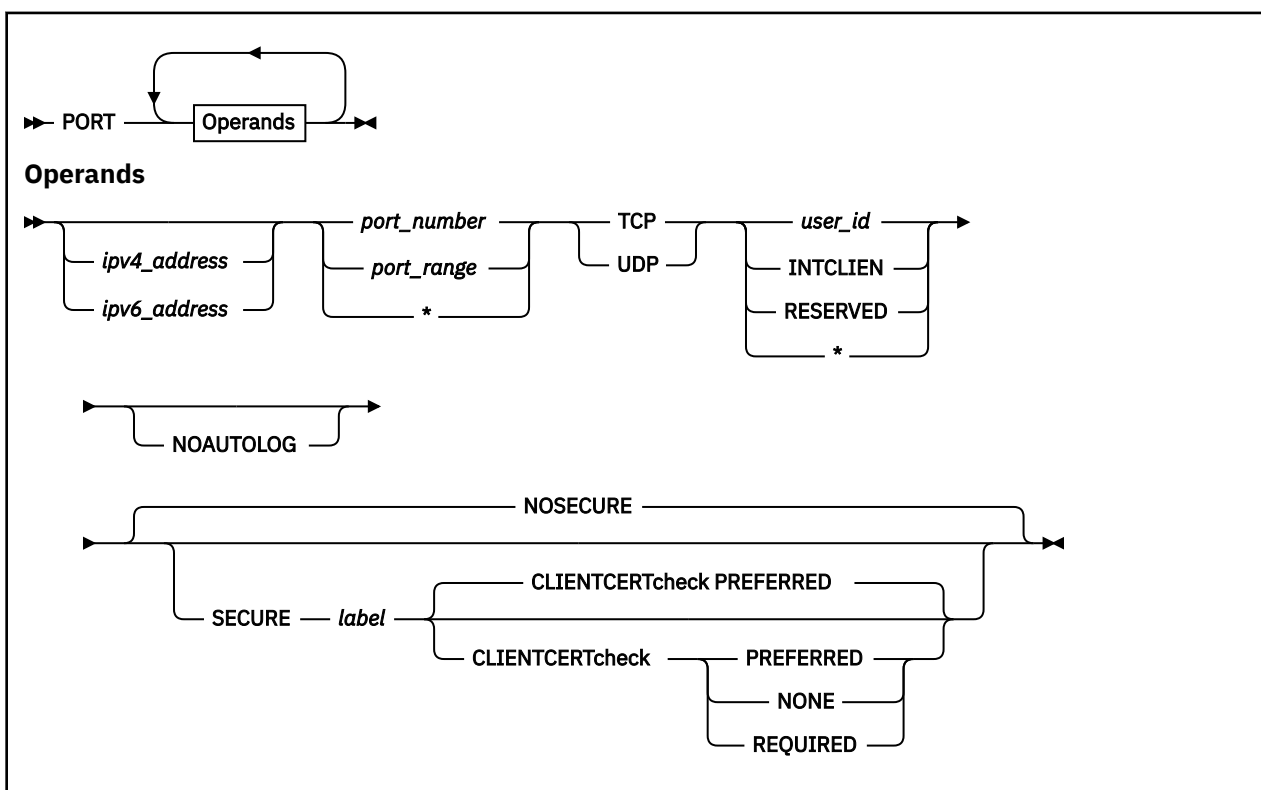
- A PERSISTCONNECTIONLIMIT value of 0 indicates there is no limit on the number of connections in persist state allowed at any given time.
- Use the NETSTAT DOS command to display the current limit of connections in persist state.

More Information

- “OBEYFILE Command” on page 644
- *z/VM: TCP/IP User's Guide* for the NETSTAT DOS and NETSTAT RESETDOS commands

PORT Statement

Use the PORT statement to reserve one port or a range of ports for a specific user, to designate one port or a range of ports as secure, or to disable or reset TCP/IP's automatic server restart function.



Operands

ipv4_address

The home IPv4 address associated with the port to be reserved. If this operand is omitted, the port is reserved for the specified userid regardless of the home address associated with a connection. Otherwise, the port is reserved for the userid only for connections associated with the specified home IPv4 address. When a userid (such as SMTP or FTPSERVE) attempts to bind to the IPv4 INADDR ANY address, the bind is intercepted by the TCP/IP stack and converted to a bind to the IPv4 address that was specified on the PORT statement for that userid. Subsequent bind processing occurs as though the userid had originally issued the bind to the IPv4 address.

ipv6_address

The home IPv6 address associated with the port to be reserved. If this operand is omitted, the port is reserved for the specified userid regardless of the home address associated with a connection. Otherwise, the port is reserved for the userid only for connections associated with the specified home IPv6 address. When a userid attempts to bind to the IPv6 unspecified address (in6addr_any), the bind is intercepted by the TCP/IP stack and converted to a bind to the IPv6 address that was specified

PORT

on the PORT statement for that userid. Subsequent bind processing occurs as though the userid had originally issued the bind to the IPv6 address.

The IPv6 address can be specified in either the preferred or compressed form.

Example:

Preferred form:

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

Compressed form:

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

port_number

The port number to be reserved.

If you specify an asterisk (*) instead of a port number, all ports to which *user_id* is permitted to bind are treated as secure. However, no connections can be made from *user_id* when this is done. If you specify an IP address, you cannot use an asterisk (*).

port_range

The range of port numbers to be reserved.

Specify two numbers separated by a single hyphen (-) and no blank spaces in the format, x-y, where x is lower than y.

TCP

UDP

The protocol that will be used on the reserved port.

user_id

INTCLIEN

The virtual machine that may use this port-protocol combination.

The special value INTCLIEN assigns the port to the internal Telnet server rather than to a client virtual machine.

Specifying an asterisk ("*") permits port sharing by all users.

A port is *shared* when more than one user can be bound to the port concurrently. Port sharing requires permission, which you grant by reserving the port exclusively for either of the following:

- A set of named users. To permit a set of named users to a port, specify the port number more than once with a different user ID each time.
- ALL users. To permit all users, specify an asterisk ("*") instead of a user ID. An asterisk instead of a virtual machine user ID specifies explicit port sharing. The designated port(s) are left open for anyone to use.

RESERVED

No connection may be established with the specified port(s). This keyword cannot be used with a specific user ID (*user_id*) or with an "*", which implies explicit port sharing.

NOAUTOLOG

Indicates that TCP/IP is not to restart *user_id* if the user stops listening to this port. Use NOAUTOLOG when the user is in the AUTOLOG list and server availability is to be under manual control. Please note that NOAUTOLOG will not prevent a restart of the client stemming from circumstances beyond the termination of the listening connection.

NOSECURE

Indicates that the port is not to be considered secure. This is the default.

SECURE

Indicates the port is secure and that any connections accepted for it will be handled according to the Secure Sockets Layer (SSL) protocol.

label

The label of the server certificate that resides in the SSL key database that is to be used in securing the subject port.

Note: The specified label can be no more than eight characters, and must be comprised of only upper case, alphanumeric characters.

CLIENTCERTCHECK NONE

A client certificate will not be requested.

CLIENTCERTCHECK PREFERRED

A client certificate will be requested. If a client certificate is not received, the connection will proceed without it. If a client certificate is received, it will be authenticated. If the certificate is not valid, the failure will be logged in the SSL server console log and the connection will continue as a secure connection protected by the server certificate.

CLIENTCERTCHECK REQUIRED

A client certificate will be requested and authenticated. If a client certificate is not received, the connection will be terminated with a fatal TLS error. If the certificate fails authentication, the handshake will fail.

Examples

1. The following example shows four servers that provide World-Wide Web services; but one, HTTPD4, is not to be monitored by TCP/IP. No other user may establish a TCP connection on port 80.

```
AutoLog
  HTTPD1 0
  HTTPD2 0
  HTTPD3 0
  HTTPD4 0
EndAutoLog

Port
  80 TCP HTTPD1
  80 TCP HTTPD2
  80 TCP HTTPD3
  80 TCP HTTPD4 NOAUTOLOG
```

2. The following example shows a group of secure ports.

```
Port
  21 TCP FTPSRV15 SECURE CERT512 ; FTP SERVER
  * TCP FTPSRV15 SECURE CERT512 ; FTP SERVER
```

In this example:

- Port 21, used by the FTPSRV15 server, is secure, and CERT512 is the certificate to be used for connections to this port.
- All ports that the FTPSRV15 server binds to will be considered secure.

Usage Notes

- RESTRICTLOWPORTS on the ASSORTEDPARMS statement is the default. Ports 1 through 1023 are protected unless the FREELOWPORTS is specified on the ASSORTEDPARMS statement. FREELOWPORTS opens all ports for access unless otherwise specified in the PORT statement.

With RESTRICTLOWPORTS, all TCP/IP applications which listen on ports 1 through 1023 must be given permission to do so. Permission is granted in one of three ways:

1. Use the PORT statement to reserve the port for the application (virtual machine), which is the preferred method. Note you can also reserve ports using port number ranges.
2. Modify the OBEY statement such that affected application virtual machine is included in the TCP/IP obey list.
3. Add FREELOWPORTS to the ASSORTEDPARMS statement. This option removes default low port protections.

- A user who is assigned a port is considered to be in the PERMIT list.
- Users in the OBEY list may access any port EXCEPT those ports specified as "Reserved".
- Any user may access a port which is explicitly shared via the "*" operand, regardless of the RESTRICTLOWPORTS setting.
- The PORT statement may appear more than once. If a syntax error is found in a PORT statement, the remainder of the statement is ignored. Multiple PORT statements previously defined or in the same profile or OBEYFILE, add to the previous port reservations. If multiple statements refer to the same port, the last statement is held to be true if the port is not reserved.

Example: Given the FREELOWPORTS setting and the following PORT statement:

```
PORT
80 TCP HTTPD1
80 UDP HTTPD1
80 TCP HTTPD2
80 UDP HTTPD2
1-1023 TCP RESERVED NOAUTOLOG
1-1023 UDP RESERVED NOAUTOLOG
23 TCP INTCLIEN
...
```

Ports 1-1023 are reserved, except for port 23, which is under the control of INTCLIEN.

- To remove the reservation for a port, the PORT statement must be deleted from the configuration file and TCP/IP must be restarted.
- The PORT statement for the internal Telnet server must specify the TCP protocol and user INTCLIEN. The port number must match the value(s) specified on the INTERNALCLIENTPARMS statement. Using the defaults, the PORT statement for the Telnet server would be

```
PORT
23 TCP INTCLIEN
```

The Telnet server automatically starts if a TCP port is reserved for INTCLIEN. To disable the Telnet server, omit or comment out this PORT statement.

- The NOAUTOLOG operand is ignored if an asterisk (*) is specified for the port number.
- If you are using a secure FTP client, you must specify an asterisk (*) for the port number. This allows any port that the client uses for the data connection to be treated as a secure port. The FTP client must use passive mode.
- If a user is no longer being autologged because MAXRESTART attempts to restart the user have been made, you can turn the autologging of the user back on by dynamically specifying the user's PORT statement without the NOAUTOLOG operand. For more information see [“MAXRESTART Statement” on page 593](#).
- Resetting the MAXRESTART counter for an individual user (via NETSTAT OBEY PORT nn TCP userid) does not affect the MAXRESTART counter of the other users, and user who share the same port number.

More Information

- [“ASSORTEDPARMS Statement” on page 529](#)
- [“AUTOLOG Statement” on page 534](#)
- [“INTERNALCLIENTPARMS Statement” on page 584](#)
- [“PERMIT Statement” on page 601](#)

PRIMARYINTERFACE Statement

Use the PRIMARYINTERFACE statement to specify which link's home address to use as the default local address.

Restriction: This statement applies to IPv4 links only.

The primary interface is the address that is inserted as the source internet address in an IP header, when communicating to a destination through an indirect route.

Only one link can be assigned as the primary interface. If multiple PRIMARYINTERFACE statements are specified, the last statement is used.

The PRIMARYINTERFACE statement has no effect on outbound traffic to a network when multiple interfaces attached to the same subnet are in use, except when the SOURCEVIPA option of the ASSORTEDPARMS statement is used. For more information about how outbound traffic is routed to a network in the presence of multiple interfaces attached to the same subnet, see [“Multiple Interface Network Support”](#) on page 512.

►► PRIMARYINTERFACE *link_name* ◄◄

Operands

link_name

The name of a link (as defined in a LINK statement) that is to be the primary interface.

Examples

The following example shows a PRIMARYINTERFACE statement that specifies a device.

```

DEVICE OSALCS1 LCS 3400 LCS
LINK LCS1 ETHERNET 0 OSALCS1

DEVICE GDLVMWEB CTC F211
LINK WEB CTC 1 VMWEB

DEVICE CTC2 CTC 0502
LINK BJACK CTC 1 CTC2

HOME
  9.227.55.27 BJACK
  9.227.44.100 LCS1
  9.227.55.9 WEB

PRIMARYINTERFACE LCS1

```

You can verify the HOME entry is the primary by using the NETSTAT HOME command. The first entry in the list will be used as the primary.

```

Home address list:

Address          Link
-----
9.227.44.100     LCS1
9.227.55.27      BJACK
9.227.55.9       WEB
Ready;

```

Usage Notes

- If you are using the PRIMARYINTERFACE statement, it must appear after the HOME statement.
- The primary interface will always appear first in the list of host addresses displayed by NETSTAT HOME.
- If the PRIMARYINTERFACE statement is not specified, then the first address in the HOME list is the default local address.

More Information

- [“Attaching network devices to the TCPIP virtual machine”](#) on page 519
- [“HOME Statement”](#) on page 579

RCBPOOLSIZE Statement

Use the RCBPOOLSIZE statement to set the initial number of raw IP control blocks (RCBs). RCBs are used to hold information about internet protocols opened by a client program, either through a RawIPOpen Pascal interface call, or a socket() call with TYPE=SOCK_RAW.



Operands

number

The initial number of RCBs in the free pool. The default is 50.

Examples

The following example shows an RCBPOOLSIZE statement specifying the default number of RCBs.

```
RCBpoolSize    50
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- Each RawIPOpen or socket() call requires one RCB, which is freed by the corresponding RawIPClose or close() call.
- The system will attempt to dynamically allocate 10% more RCBs any time the RCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many RCBs your system is using. To avoid dynamic allocation of RCBs during operation, use the RCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

RESTRICT Statement

Use the RESTRICT statement to identify users who are prohibited from using TCP/IP services.

The first RESTRICT statement of each configuration file processed replaces the existing RESTRICT list with the new one. Subsequent RESTRICT statements in the same file add users to the list.



Operands

user_id

A user or group of users that are prohibited from using TCP/IP services.

An asterisk (wild-card character) can be used at the end of a user ID to match any character. For example, NONIBM* matches all user IDs starting with NONIBM, such as NONIBM, NONIBM1, NONIBMA, and NONIBMYZ.

Examples

The following example shows how to deny TCP/IP services to user BADGUY and to any user ID beginning with the characters "VEND".

```
Restrict
VEND*
BADGUY
EndRestrict
```

Usage Notes

- The ENDRESTRICT statement ends the RESTRICT statement. If ENDRESTRICT is omitted, all subsequent statements are interpreted as user IDs.
- If a user is specified on both the PERMIT and RESTRICT statements, the user will not be able to use TCP/IP services.
- If a restricted user attempts to use TCP/IP services, the message

```
date time Unauthorized TCP/IP access attempt by userid
```

is logged by the TCPIP virtual machine in the stack console or in the trace file (whichever is active) and to a special authorization audit file named TCPIP AUTHLOG A.

You can override the audit file specification by including an appropriate :Authlog. tag entry in a customized DTCPARMS file definition for the TCP/IP server.

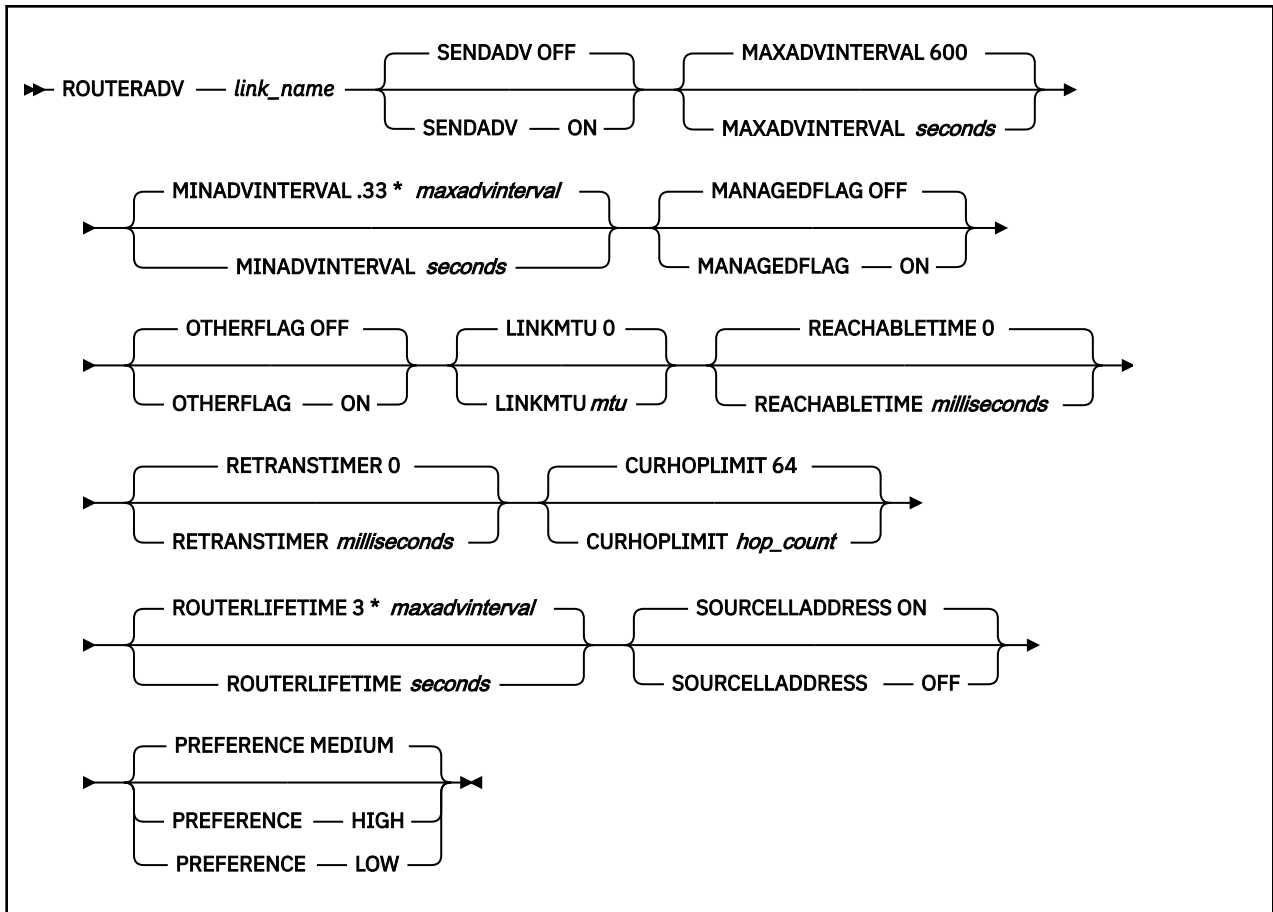
More Information

- [“PERMIT Statement” on page 601](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)

ROUTERADV Statement

Use the ROUTERADV statement to specify whether router advertisements should be sent on a link, and, if so, to specify for that link the minimum and maximum time between sending unsolicited router advertisements as well as other information that will be included in router advertisements on that link.

Note: Each subsequent ROUTERADV statement specified for the same link replaces all information used for router advertisements on the link, so, if default values are undesirable, each parameter must be specified on each subsequent ROUTERADV statement. ROUTERADVPREFIX information for a given link remains unchanged when a ROUTERADV statement is processed.



Operands

link_name

Specifies the name of the IPv6 link (defined in a previous LINK statement) associated with this router advertisement information.

SENDADV ON | OFF

Determines whether periodic router advertisements are sent and whether router solicitations are responded to on this link. Specify ON to enable router advertisements on the link. The default is OFF.

MAXADVERTINTERVAL *seconds*

Specifies the maximum time, in seconds, between sending unsolicited multicast router advertisements on this link. Specify *seconds* as an integer in the range of 4 through 1800. The default is **600** seconds.

MINADVERTINTERVAL *seconds*

Specifies the minimum time, in seconds, between sending unsolicited multicast router advertisements on this link. The value of *seconds* must be an integer in the range of 3 seconds (the minimum) to .75 times the value of MAXADVERTINTERVAL (the maximum). The default is .33 * MAXADVERTINTERVAL in seconds.

MANAGEDFLAG ON | OFF

Specifies whether hosts should use the administered (stateful) protocol for address autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration. The use of this flag is described in RFC 2462. The default is OFF.

OTHERFLAG ON | OFF

Specifies whether hosts should use the administered (stateful) protocol for autoconfiguration of other (non-address) information. The use of this flag is described in RFC 2462. When the MANAGEDFLAG is set ON, the OTHERFLAG must also be ON. (The default for OTHERFLAG is OFF.) If MANAGEDFLAG is ON and OTHERFLAG is OFF, a warning message is issued and OTHERFLAG is implicitly changed to ON.

LINKMTU *mtu*

Specifies the value to be placed in MTU options sent in router advertisements on this link. MTU options allow hosts to use the same link MTU when the link MTU is not well known. If zero, no MTU options will be sent. If specified (not zero), *mtu* must be an integer in the range of 576 through the maximum MTU allowed on a given link. The default is **0**.

REACHABLETIME *milliseconds*

Specifies the time, in milliseconds, that a node assumes a neighbor is reachable after having received a reachability confirmation. The setting is used by the Neighbor Unreachability Detection algorithm described in section 7.3 of RFC 2461. A value of zero means unspecified. If specified (not zero), *milliseconds* must be an integer in the range of 1 through 3,600,000 (1 hour). The default is **0**.

RETRANSTIMER *milliseconds*

Specifies the time, in milliseconds, between retransmitted neighbor solicitation messages. The setting is used by address resolution and the Neighbor Unreachability Detection algorithm (see sections 7.2 and 7.3 of RFC 2461). A value of zero means unspecified. If specified (not zero), *milliseconds* must be an integer in the range of 1 through 2,147,483,647. The default is **0**. If an integer greater than 2,147,483,647 is specified, the value is ignored and the RETRANSTIMER is set to the maximum value (2,147,483,647).

CURHOPLIMIT *hop_count*

Specifies the default value to be placed in the hop count field of the IP header for outgoing IP packets. A value of zero means unspecified. If specified (not zero), *hop_count* must be an integer in the range of 1 through 255. The value should be set to the current diameter of the internet (the average shortest path between two points in the internet). The default is 64.

ROUTERLIFETIME *seconds*

Specifies the lifetime associated with the default router in seconds. A value of zero indicates that the router is not a default router and should not appear on the default router list. The lifetime applies only to the router's usefulness as a default router. The lifetime does not apply to information contained in other message fields or options. Options that need time limits for their information include their own lifetime fields. If not zero, *seconds* must be an integer in the range of MAXADVINTERVAL through 9000 seconds. The default is 3 times the value of MAXADVINTERVAL.

SOURCEADDRESS ON | OFF

Specifies whether a source link-layer address option should be included in router advertisements on this link. Setting ignored when *link_name* is a HiperSockets link (type QDIOIP). The default is ON.

PREFERENCE HIGH | MEDIUM | LOW

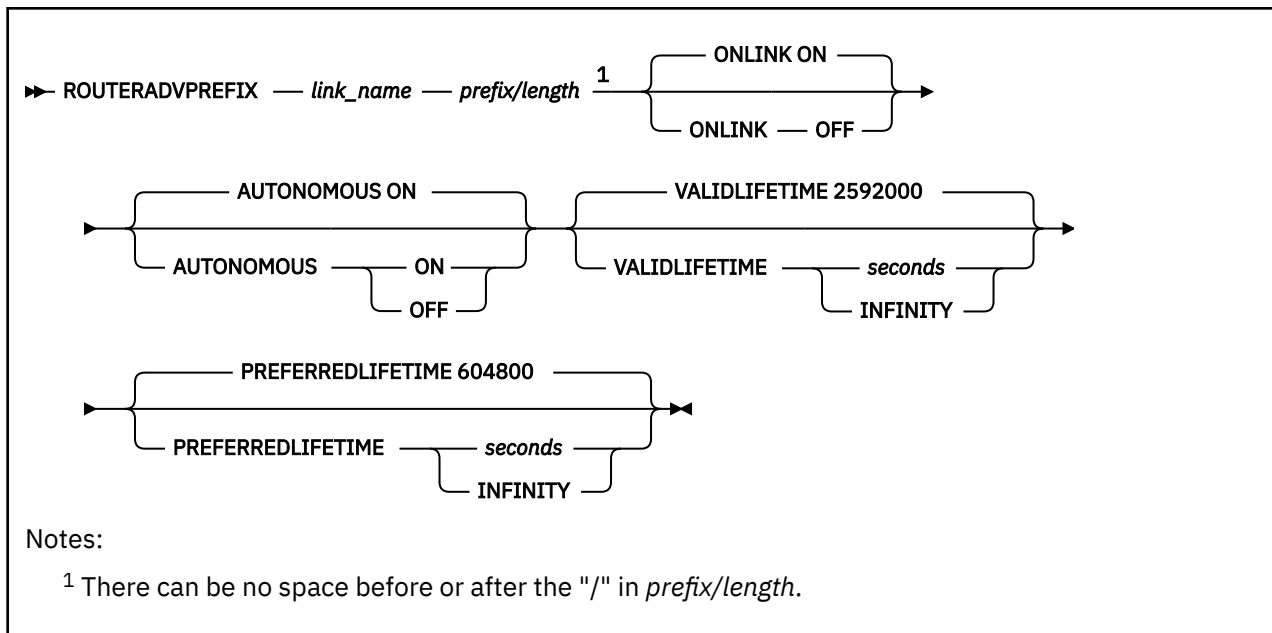
Determines whether the router advertisements are sent with a HIGH, MEDIUM, or LOW preference value, which indicates whether the route should be preferred over other default routers. See RFC 4191 for detailed information. The default is MEDIUM.

Usage Notes

- When you configure a link as an advertising interface (using SENDADV ON), the system prefers configured Router Advertisement information from ROUTERADV and ROUTERADVPREFIX statements over information from received Router Advertisements.
- You can specify optional parameters on the ROUTERADV statement in any order.
- You can check the current configuration with the NETSTAT CONFIG ROUTERADV command.

ROUTERADVPREFIX Statement

Use the ROUTERADVPREFIX statement to specify prefixes that will be included in router advertisements on specific links. These prefixes are used for autonomous address configuration and on-link determination.



Operands

link_name

Specifies the name of the IPv6 link (defined in a previous `LINK` statement) associated with this router advertisement prefix.

prefix

Specifies an IPv6 address or prefix to be used for on-link determination or autonomous address configuration on this link. The prefix cannot be the link-local prefix.

length

Specifies the number of bits in the prefix that are valid. The bits in the prefix after the prefix length will be initialized to zero. Specify *length* as an integer in the range of 1 through 128.

ONLINK ON | OFF

Specifies whether the given prefix can be used for on-link determination. `ON` indicates that the prefix can be used for on-link determination. `OFF` indicates that the router advertisement makes no statement about the on-link or off-link properties of the prefix. For instance, the prefix might be used for address configuration with some of the addresses belonging to the prefix being on-link and others being off-link. The default is `ON`.

AUTONOMOUS ON | OFF

Specifies whether the given prefix can be used for autonomous address configuration as specified in RFC 2462. The default is `ON`. Prefixes configured to be autonomous (`AUTONOMOUS ON`) must have a prefix length of 64.

VALIDLIFETIME seconds

Specifies the length of time, relative to the time the router advertisement is sent, that

- The prefix for the purpose of on-link determination remains valid
- Addresses generated from the prefix through stateless address auto-configuration remain valid.

Specify *seconds* as an integer in the range of 0 through 2,147,483,647, or use the keyword `INFINITY` for infinity. The default is 2,592,000 seconds (30 days). If an integer greater than 2,147,483,647 is specified, the value is ignored and the `VALIDLIFETIME` is set to the maximum value (2,147,483,647).

The value for `VALIDLIFETIME` must be greater than or equal to the value for `PREFERREDLIFETIME`.

PREFERREDLIFETIME seconds

Specifies the length of time, in seconds (relative to the time the router advertisement is sent), that addresses generated from the prefix via stateless address autoconfiguration remain preferred. Specify *seconds* as an integer in the range of 0 through 2,147,483,647, or use the keyword `INFINITY`.

for infinity. The default is 604,800 seconds (7 days). If an integer greater than 2,147,483,467 is specified, the value is ignored and the PREFERREDLIFETIME is set to the maximum value (2,147,483,647).

Usage Notes

- You do not need to specify all prefix information for all links in each OBEYFILE. However, if a ROUTERADVPREFIX statement exists in an OBEYFILE for a specific link, all current prefix information for that link is deleted. Therefore, if a ROUTERADVPREFIX statement exists for a specific link in an OBEYFILE, all prefixes for that link must be configured in the OBEYFILE using separate ROUTERADVPREFIX statements.
- If you give multiple ROUTERADVPREFIX statements for the same prefix/prefix length pair in the same configuration file, information from the last statement is used.
- If EQUALCOSTIPV6MULTIPATH support is enabled and a direct prefix route already exists for the prefix in the ROUTERADVPREFIX statement, the LINKMTU specified must match the packet size of the route already in the routing table. If the LINKMTU does not match, the ROUTERADVPREFIX statement will not change the packet size in the IPv6 routing table. In order to change the packet size (LINKMTU) in this case, a ROUTERADVPREFIX statement with a VALIDLIFETIME of zero for the prefix must be obeyed to delete the route. Then, a ROUTERADVPREFIX statement with the new LINKMTU must be obeyed to add the route back in to the IPv6 routing table.
- You can check the current configuration with the NETSTAT CONFIG ROUTERADV command.

SCBPOOLSIZE Statement

Use the SCBPOOLSIZE statement to set the initial number of socket control blocks (SCBs). SCBs are used to hold information about TCP connections and UDP ports.



Operands

number

The initial number of SCBs in the free pool. The default is 256.

Examples

The following example shows an SCBPOOLSIZE statement specifying the default number of SCBs.

```
SCBpoolSize 256
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system will attempt to dynamically allocate 10% more SCBs any time the SCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many SCBs your system is using. To avoid dynamic allocation of SCBs during operation, use the SCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- Running out of SCBs prevents the opening of new TCP connections and UDP ports.

More Information

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

SCREEN Statement

Use the SCREEN statement to direct TCP/IP stack run-time trace output to the console and close any current trace file.

```
➡ SCREEN ➡
```

Operands

The SCREEN statement has no operands.

More Information

- [“TRACE Statement” on page 623](#)
- [“NOSCREEN Statement” on page 596](#)

SKCBPOOLSIZE Statement

Use the SKCBPOOLSIZE statement to set the initial number of socket interface control blocks (SKCBs). SKCBs are used to hold information about communication endpoints established with the socket(), accept(), or takesocket() calls.

```
➡ { SKCBPOOLSIZE — 256
    SKCBPOOLSIZE — number } ➡
```

Operands

number

The initial number of SKCBs in the free pool. The default is 256.

Examples

The following example shows an SKCBPOOLSIZE statement specifying the default number of SKCBs.

```
SKCBpoolSize 256
```

Usage Notes

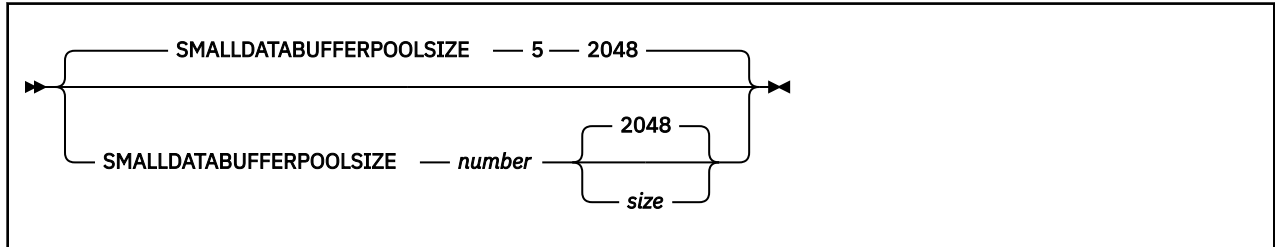
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- Each communication endpoint requires one SKCB, which is freed by a close() call.
- The system will attempt to dynamically allocate 10% more SKCBs any time the SKCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many SKCBs your system is using. To avoid dynamic allocation of SKCBs during operation, use the SKCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

SMALLDATABUFFERPOOLSIZE Statement

Use the SMALLDATABUFFERPOOLSIZE statement to set the initial number and size of small data buffers. Small data buffers usually hold 2,048 bytes of data, but may be configured to hold up to 4,096 bytes of data.



Operands

number

The initial number of data buffers in the small data buffer pool. The minimum and default is 5.

size

The size of each small data buffer (in bytes). The default size is 2048. Only one of the following values or their alternatives (shown in parenthesis) can be specified:

- 2048 (2K)
- 3072 (3K)
- 4096 (4K)

Examples

The following example shows a SMALLDATABUFFERPOOLSIZE statement specifying the number of small data buffers to be 1200.

```
SmallDataBufferPoolSize 1200
```

Usage Notes

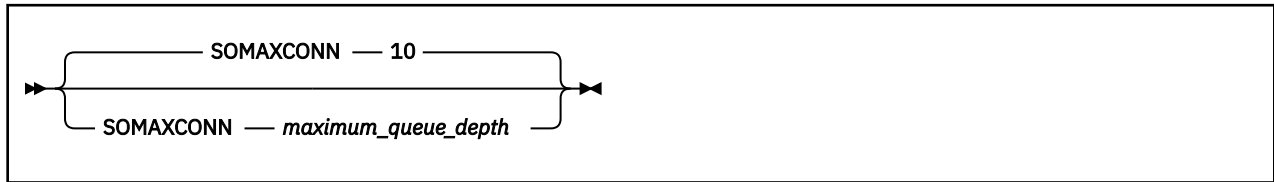
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system will attempt to dynamically allocate 10% more small data buffers any time the small data buffer free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many small data buffers your system is using. To avoid dynamic allocation of small data buffers during operation, use the SMALLDATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [“DATABUFFERPOOLSIZE Statement” on page 537](#)
- [“TINYDATABUFFERPOOLSIZE Statement” on page 621](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

SOMAXCONN Statement

Use the SOMAXCONN statement to specify the maximum length for a connection request queue created by the socket call `listen()`.



Operands

maximum_queue_depth

The maximum number of pending connection requests queued for any listening socket. The default is 10.

Examples

The following example shows a SOMAXCONN statement specifying the maximum number of pending connection requests queued for any listening socket to be 10.

```
SOMAXCONN 10
```

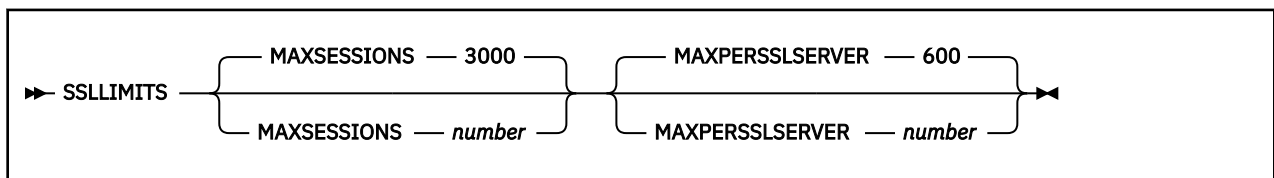
Usage Notes

- The maximum queue depth number can be any integer that stores in a fullword, though most TCP/IP implementations code a value in the range of 5 to 10.
- The maximum queue depth number is the maximum depth for any listening stream socket, but the programmer can specify a shorter queue length on each `listen()` socket call.
- TCP/IP determines the maximum `listen()` backlog queue length for each socket by choosing the lower of two values: either the backlog argument specified on the `listen()` or the value specified on the SOMAXCONN statement in PROFILE TCPIP. Thus, if you specify the SOMAXCONN constant in C socket programs to determine what the acceptable maximum backlog queue length is, TCP/IP might not use the value in SOCKET.H. For example, if the SOMAXCONN value in SOCKET.H is 22, the PROFILE TCPIP has SOMAXCONN 10, and you specify SOMAXCONN as the backlog argument on `listen()`, TCP/IP uses the lower value, 10.

To prevent a mismatch in values, remember to change the header file to reflect the value specified for the TCP/IP server in the SOMAXCONN statement.

SSLIMITS Statement

Use the SSLIMITS statement to specify the total number of secure connections allowed and the connection limit for each SSL server.



Operands

MAXSESSIONS

The maximum number of secure connections allowed across all SSL servers. This is an integer between 1 and 2,147,483,647. The default is 3000. It is recommended that this number not exceed 5000.

MAXPERSSLSERVER

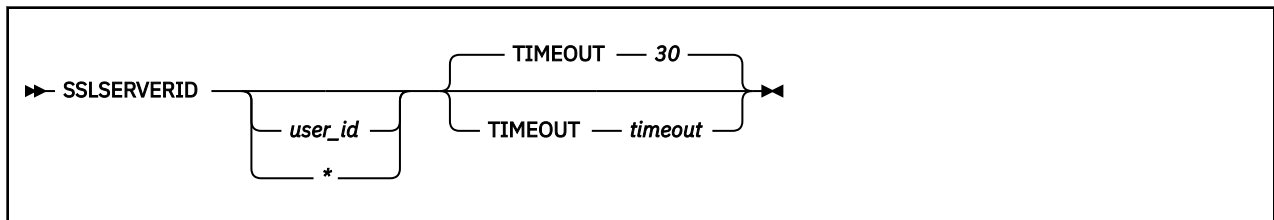
The maximum number of connections that can flow through a single SSL server. This is an integer between 1 and 1000. The default is 600.

Usage Notes

- The maximum number of SSL servers required is obtained by dividing MAXSESSIONS by MAXPERSSLSERVER. It is recommended that these values divide evenly.
- If the MAXPERSSLSERVER value is greater than the MAXSESSIONS value, the MAXPERSSLSERVER will be changed to the MAXSESSIONS value as long as MAXSESSIONS is less than 1000. If MAXSESSIONS is greater than 1000, the MAXPERSSLSERVER value will be set at 1000.

SSLSERVERID Statement

Use the SSLSERVERID statement to force autologging of an SSL server pool before all other servers in the AUTOLOG list. This statement should be used if any protocol servers used by your installation are configured to provide Dynamic SSL/TLS connection support.



Operands

user_id

User ID of the SSL server that is to be autologged. An asterisk (*) indicates that a pool of SSL user IDs (servers) are used to provide secure connection support. Detailed information about this server pool is defined in the DTCPARMS file.

timeout

Number of seconds to wait for SSL server initialization to complete, before the other virtual machines (identified by the AUTOLOG statement) are autologged. The *timeout* minimum is 0, the maximum is 120, and the default is 30 seconds.

Examples

The following is an example of an SSLSERVERID statement. This statement will cause an autolog of the server pool identified and defined in a DTCPARMS file for the subject TCP/IP stack server. Once the SSL server pool has initialized, or two minutes have elapsed, the other virtual machines associated with this stack server will be autologged.

```
SSLSERVERID * TIMEOUT 120
```

Usage Notes

- All SSL server user IDs will be AUTOLOGGED. When '*' is specified, the first server of the SSL server pool is autologged before any other pool servers.

START

- It is recommended that the TCP/IP server be allowed to manage initialization of the SSL server pool through the use of the SSLSERVERID statement only. The use of another mechanism for this purpose (such as the AUTOLOG1 virtual machine) might impact the ability for secured connections to be established as required for your installation.

More Information

- [“AUTOLOG Statement” on page 534](#)

START Statement

Use the START statement to start a device that is currently stopped. This statement is usually specified at the end of the configuration file.

```
➤ START — device_name ➤
```

Operands

device_name

The name of the device to start. This must be the same *device_name* specified in a DEVICE statement.

Examples

The following example shows START statements that start LCS devices.

```
Start LCS1  
Start LCS2
```

Usage Notes

- Each device to be started requires a separate START statement.
- The START statement can also be used in an obey file to start:
 - A newly-defined device;
 - A device stopped with the STOP statement;
 - A device that failed to initialize when TCP/IP started.

More Information

- [“Attaching network devices to the TCPIP virtual machine” on page 519](#)
- [“STOP Statement” on page 618](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)

STOP Statement

Use the STOP statement in an obey file to stop a device that is currently started.

```
➤ STOP — device_name ➤
```

Operands

device_name

The name of the device to be stopped. This must be the same *device_name* specified in a DEVICE statement.

Usage Notes

- Storage used by the device driver is not freed; it is reused the next time the device is started.
- Each device to be stopped requires a separate STOP statement.

More Information

- [“Attaching network devices to the TCPIP virtual machine” on page 519](#)
- [“START Statement” on page 618](#)
- [“Changing the TCP/IP Configuration with the OBEYFILE Command” on page 644](#)

SYSCONTACT Statement

Use the SYSCONTACT statement to specify the value of the SNMP object *sysContact*. SYSCONTACT is the textual identification of the contact person for this host, together with information about how to contact them.

```
➤ SYSCONTACT — contact_info — ENDSYSCONTACT ➤
```

Operands

contact_info

Up to 255 characters of name and contact information for the administrator of this host. TCP/IP converts the specified text to upper case before storing it.

Examples

The following example shows a SYSCONTACT statement that defines the contact person and phone number responsible for this host.

```
SysContact
  Main Operator (555-1234)
EndSysContact
```

Usage Notes

The ENDSYSCONTACT statement specifies the end of the SYSCONTACT information.

More Information

- [Chapter 14, “Configuring the SNMP Servers,” on page 439](#)

SYSLOCATION Statement

Use the SYSLOCATION statement to specify the value of the SNMP object *sysLocation*. SYSLOCATION is the physical location of this host (for example, *telephone closet, 3rd floor*).

```
➤ SYSLOCATION — location_info — ENDSYSLOCATION ➤
```

Operands

location_info

Up to 255 characters describing the physical location of this node. TCP/IP converts the specified text to upper case before storing it.

Examples

The following example shows a SYSLOCATION statement that defines the physical location of the host.

```
SysLocation
  First floor Computer Room
EndSysLocation
```

Usage Notes

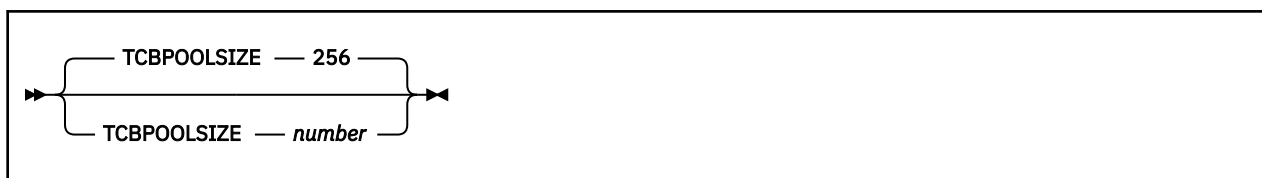
The ENDSYSLOCATION statement specifies the end of the SYSLOCATION information.

More Information

- [Chapter 14, “Configuring the SNMP Servers,” on page 439](#)

TCBPOOLSIZE Statement

Use the TCBPOOLSIZE statement to set the initial number of TCP control blocks (TCBs). TCBs are used to hold information about TCP connections.



Operands

number

The initial number of TCBs in the free pool. The default is 256; the minimum is 6.

Examples

The following example shows a TCBPOOLSIZE statement setting the number of TCBs to the default of 256.

```
TCBpoolSize 256
```

Usage Notes

- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system will attempt to dynamically allocate 10% more TCBs any time the TCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many TCBs your system is using. To avoid dynamic allocation of TCBs during operation, use the TCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- The TCB pool can be expanded to either 10,000 TCBs or twice its initial size, whichever value is greater.

More Information

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

TIMESTAMP Statement

Use the TIMESTAMP statement to select when to write time stamps in the trace and console files, in order to show when events happened.



Operands

num_msg

A number that indicates when time stamps are displayed with a message. If 0 is specified, no time stamps are displayed. Otherwise, time stamps are generated every *num_msg* messages.

PREFIX

Specifies that a time stamp prefixes every message. This is the default.

Examples

The following example generates a time stamp each time 5 messages have been displayed.

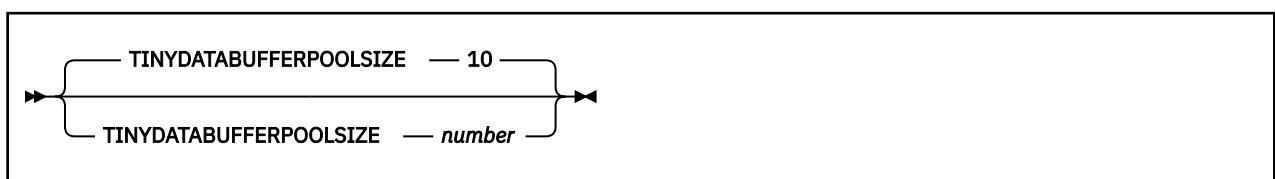
```
TIMESTAMP 5
```

More Information

- [“TRACE Statement”](#) on page 623

TINYDATABUFFERPOOLSIZE Statement

Use the TINYDATABUFFERPOOLSIZE statement to set the initial number of tiny data buffers. Tiny data buffers hold 256 bytes of data.



Operands

number

The initial number of data buffers in the tiny data buffer pool. The minimum and default is 10.

Examples

The following example shows a TINYDATABUFFERPOOLSIZE statement specifying the number of tiny data buffers to be 500.

```
TinyDataBufferPoolSize 500
```

Usage Notes

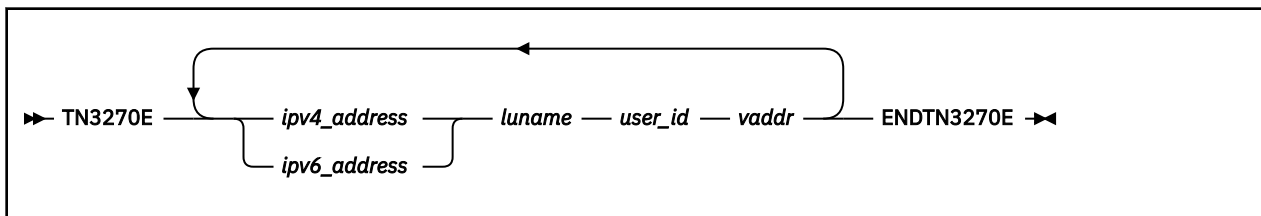
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system will attempt to dynamically allocate 10% more tiny data buffers any time the tiny data buffer free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many tiny data buffers your system is using. To avoid dynamic allocation of tiny data buffers during operation, use the TINYDATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- If there are data buffers in the tiny data buffer pool, TCP/IP will use them for applications that exchange small amounts of data, such as the FTP and Telnet servers.

More Information

- [“DATABUFFERPOOLSIZE Statement” on page 537](#)
- [“SMALLDATABUFFERPOOLSIZE Statement” on page 615](#)
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

TN3270E Statement

Use the TN3270E statement to define client IP addresses and logical unit names that might establish printer sessions and associate a user identifier and virtual address with the session.



Operands

ipv4_address

ipv6_address

The IPv4 or IPv6 address of the client who is permitted to establish a TN3270E printer session.

luname

The logical unit name the client must provide.

user_id

The user identifier of the virtual machine to be associated with the printer session (for example, RSCS).

vaddr

The virtual address in the aforementioned virtual machine that is associated with the printer session. The virtual address must be within the range 1-FFFF.

Examples

The following example allows two TN3270E printer sessions to be established.

```

TN3270E
  9.130.58.29 JMARSH RSCS 360
  9.130.67.224 REFSNID RSCS 361
ENDTN3270E

```

Usage Notes

- If the IP address and LU name supplied by a client who is establishing a TN3270E printer session match a specification in the TN3270E statement, the associated user identifier and virtual device address are provided to the Printer Management Exit.
- When a printer session is accepted, a logical printer device is created to represent it.
- A TN3270E printer management exit must be provided to connect the logical printer device to a virtual machine that can create and transmit 3270 data streams to it. See the [z/VM: TCP/IP Programmer's Reference](#) for more information about this exit.
- If the OBEYFILE command is used and includes the TN3270E statement, the entire set of printer session definitions is replaced. Existing printer sessions are not affected.
- If a syntax error is found in a TN3270E statement, the remainder of the statement is ignored. Subsequent TN3270E statements in the same profile or OBEYFILE are processed.
- The same virtual device address cannot be specified more than once for a given user ID.
- The virtual device address specified must not be in use by the user ID specified.

More Information

- [“INTERNALCLIENTPARMS Statement” on page 584](#)

TRACE Statement

Use the TRACE statement to establish the list of internal TCP/IP stack processes for run-time tracing. The TRACE statement is intended for debugging, in consultation with the IBM Support Center.



Operands

MOST

Enables run-time tracing for all processes except INITIALIZE, SCHEDULER, and TIMER.

process_name

Enables run-time tracing for the named process. For valid process names, see [Table 46 on page 624](#).

Usage Notes

- If no process name or an invalid process name is specified, detailed tracing is enabled for the processes in the MOST process name alias.
 - The SCREEN configuration statement directs trace messages to the console. The NOSCREEN statement directs them to the file selected by the FILE statement.
- Tracing TCP/IP processes can (and often will) produce a large amount of data. If trace messages are directed to the trace file and the disk containing the trace file becomes full, trace output is terminated.
- To disable run-time tracing, use the NOTRACE configuration statement.
 - If a TRACEONLY statement is used, the trace is restricted to the processes that are related to the selected users, devices, or IP addresses.

More Information

- [“FILE Statement” on page 565](#)
- [“LESSTRACE Statement” on page 592](#)
- [“MORETRACE Statement” on page 595](#)
- [“NOTRACE Statement” on page 597](#)
- [“SCREEN Statement” on page 614](#)
- [“NOSCREEN Statement” on page 596](#)
- [“TRACEONLY Statement” on page 625](#)
- [z/VM: TCP/IP Diagnosis Guide](#)

The following table includes a list of TCP/IP process names. These names are valid for the TRACE, LESSTRACE, MORETRACE, and NOTRACE statements.

Table 46. TCP/IP Process Names

Process Names		
ALL	IUCV	ROUND-TRIP
ARP	IUCVGREETER	SCHEDULER
CCS	IUCVHANDLER	SECURITY
CONGESTION	IUCVSIGNON	SHUT-DOWN
CONSISTENCY-CHECKER	MONITOR	SNMPDPI
CTC	MOST	SOCKET
DENIALOFSERVICE	MULTICAST	STATUS-IN
DROPPED	NED	STATUS-OUT
DYNROUTING	NO-PROCESS	TABLE
EQDIO-CONTROL	NONE	TCP
EQDIO-DATA	NOTIFY	TCP-DOWN
EXTERNAL-HANDLER	OSD	TCP-IP
FPSM	PACKET	TCP-REQUEST
HANDLERS	PARSE-TCP	TCP-UP
ICMP	PCCA	TELNET
IGMP	PING	TIMER
INITIALIZE	QDIO	TOIUCV
IO-HANDLER	RAWIP	TRACETABLE
IOCTLROUTE	RAWIPREQUEST	UDP
IP-DOWN	RAWIPUP	UDPREQUEST
IP-REQUEST	RETRANSMIT	UDPUP
IP-UP	REXMIT	

Notes:

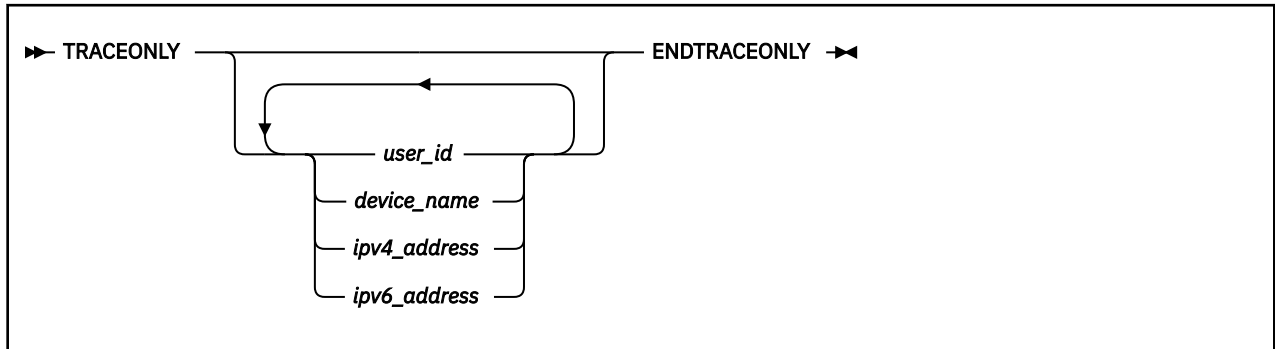
1. For process name SECURITY, a log entry is recorded whenever the TCP/IP configuration or z/VM system configuration is changed through a NETSTAT or OBEYFILE command.
 - When used with TRACE, authorization failures also generate log entries.
 - When used with MORETRACE, authorization failures and the NETSTAT CP QUERY and NETSTAT CP INDICATE commands also generate log entries.

The log entry includes the issuer of the command and any relevant data returned by the command. In addition, the OBEYFILE and NETSTAT OBEY commands generate a message sent to the users on the INFORM list.

2. Process names ALL, MOST, NONE, and NO-PROCESS are aliases. The MOST process alias includes all processes except INITIALIZE, SCHEDULER, and TIMER. The alias ALL includes all processes. TRACE NO-PROCESS and TRACE NONE turn off all tracing.
3. Hyphens (-) in process names can be omitted.
4. For process name OSD, make sure the VSWITCH CONTROLLER statement uses the FAILOVER_DISABLED option.

TRACEONLY Statement

Use the TRACEONLY statement to restrict TCP/IP stack tracing to particular users, devices, or IP addresses. It is intended for debugging, in consultation with the IBM TCP/IP support group.



Operands

user_id

The user identifier of the client you want to trace. In order for a selection to be effective, the client must be either active or included in the PORT, OBEY, INFORM, or PERMIT list.

device_name

The name of the device you want to trace.

ipv4_address

The IPv4 address you want to trace, in dotted decimal form.

ipv6_address

The IPv6 address you want to trace, in either the preferred or compressed form.

Example:

Preferred form:

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

Compressed form:

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

Examples

1. The following example shows how to enable selective tracing for user FTPSERVE.

```
FILE REQUEST TRACE A
TRACEONLY FTPSERVE ENDTRACEONLY
TRACE TCPREQUEST
```

2. The following example terminates selective tracing.

```
SCREEN
NOTRACE
TRACEONLY ENDTRACEONLY
```

Usage Notes

- Specify the processes to be traced using the TRACE, MORETRACE, and LESSTRACE statements. When there is activity-related to the selected users, devices, or IP addresses, the trace is enabled. Otherwise, the trace is disabled.

- Certain processes are not relevant for certain entities. For example, the CTC process is relevant to device tracing; enabling a selective trace for a user will not produce output from the CTC process, even if the process is among those being traced.
- Use the NOTRACE statement to turn off selective tracing and, if the trace is being written on disk, close the output file.
- To turn off selective tracing and enable tracing for the entire stack, specify the TRACEONLY statement with no operands.
- A maximum of 64 IPv4 and 64 IPv6 addresses can be selected for tracing at one time.
- When TRACEONLY is used in conjunction with the PACKETTRACESIZE statement to produce TRSOURCE packet trace data, only the *device_name* operands will result in data being collected.

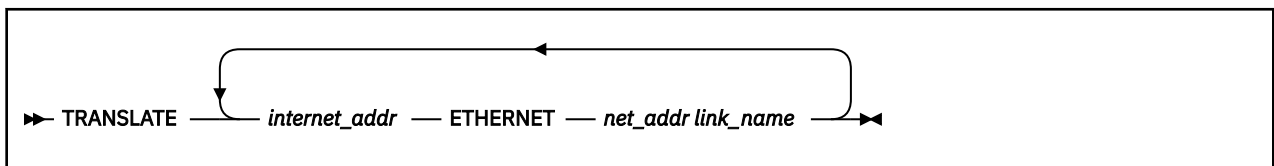
More Information

- [“FILE Statement” on page 565](#)
- [“INFORM Statement” on page 584](#)
- [“LESSTRACE Statement” on page 592](#)
- [“MORETRACE Statement” on page 595](#)
- [“NOSCREEN Statement” on page 596](#)
- [“NOTRACE Statement” on page 597](#)
- [“PACKETTRACESIZE Statement” on page 598](#)

TRANSLATE Statement

Use the TRANSLATE statement to indicate the relationship between an IP address and the network address on a specified link. You can use the TRANSLATE statement, with some limitations, for Ethernet hosts that do not support ARP.

Restriction: This statement applies to IPv4 links only.



Operands

internet_addr

The IP address in dotted decimal form for which a translation is specified.

ETHERNET

Indicates that *net_addr* is an Ethernet address.

net_addr

The network address corresponding to *internet_addr* and *link_name*. The format depends on the network type.

- For ETHERNET, specify a 12-digit hexadecimal MAC address of the remote adapter. The remote host is assumed to use network headers in DIX Ethernet format, not 802.3 format.

link_name

A network link name (from a LINK statement). The specified *internet_addr* is translated to the specified *net_addr* only when sending on this link. You can include multiple TRANSLATE statement entries for the same *internet_addr* with different *link_names*.

Usage Notes

- The first TRANSLATE statement of each configuration file processed replaces the internal translation tables (the ARP table), including information dynamically added by ARP, with the new information. Subsequent TRANSLATE statements in the same file add entries to the table. The exception to this is that entries which appear in a corresponding HOME statement are not deleted.

Note: The TRANSLATE statement is effective on a per link basis. You do not have to code a TRANSLATE statement if you want the functional MAC address, as it is the default method.

UCBPOOLSIZE Statement

Use the UCBPOOLSIZE statement to set the initial number of UDP control blocks (UCBs). UCBs are used to hold information about open UDP ports.



Operands

number

The initial number of UCBs in the free pool. The default is 100.

Examples

The following example shows a UCBPOOLSIZE statement setting the number of UCBs to the default of 100.

```
UCBpoolSize 100
```

Usage Notes

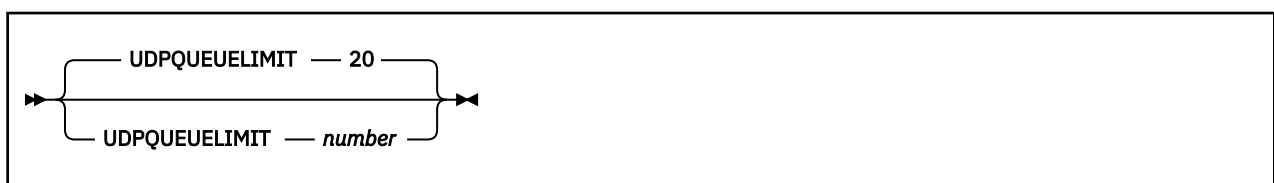
- If storage cannot be obtained for the number of pool elements requested, TCP/IP attempts to allocate 5% of that number. If it is successful in allocating 5%, initialization continues using the reduced pool size. Based on demand, dynamic allocation increases the pool size as necessary.
- The system will attempt to dynamically allocate 10% more UCBs any time the UCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many UCBs your system is using. To avoid dynamic allocation of UCBs during operation, use the UCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

More Information

- [z/VM: TCP/IP User's Guide](#) for the NETSTAT command

UDPQUEUELIMIT Statement

Use the UDPQUEUELIMIT statement to define the maximum number of incoming datagrams that can be queued on a UDP port. If a value of 0 is specified, the number of incoming datagrams is unlimited.



Operands

number

The maximum number of incoming datagrams that can be queued on a UDP port.

Examples

1. The following example shows a UDPQUEUELIMIT statement specifying the maximum number of incoming datagrams that can be queued on a UDP port to be 1000:

```
UDPQUEUELIMIT 1000
```

2. The following example shows a UDPQUEUELIMIT statement specifying that no limit is applied to the number of incoming datagrams that can be queued on a UDP port:

```
UDPQUEUELIMIT 0
```

Usage Notes

- A UDPQUEUELIMIT of 0 indicates that there is no limit to the number of incoming datagrams that can be queued on a UDP port.
- The UDPQUEUELIMIT statement takes precedence over the NOUDPQUEUELIMIT option on the ASSORTEDPARMS statement.
- Use the NETSTAT CONFIG UDPQUEUELIMIT command to display the current limit.

More Information

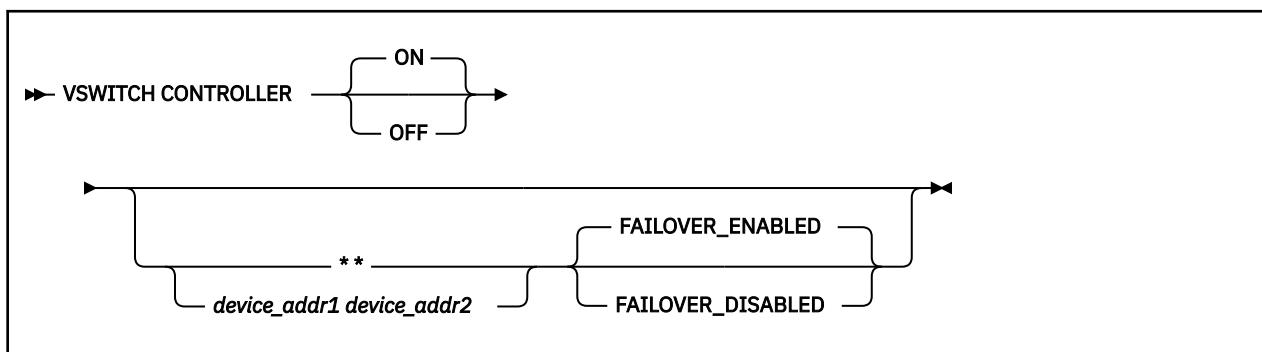
- “OBEY Statement” on page 597
- [z/VM: TCP/IP User's Guide](#) for the NETSTAT CONFIG command

VSWITCH CONTROLLER Statement

Use the VSWITCH CONTROLLER statement to specify whether a stack is available to control a CP-defined virtual switch's connection to a real LAN segment through an OSA-Express adapter. When a virtual switch is defined, CP uses a z/VM TCP/IP stack to control its interface to the network through an OSA-Express device. The VSWITCH CONTROLLER statement allows the z/VM TCP/IP stack to define where three or more control devices are attached. A controller can also manage a connection to a HiperSockets device defined as a Bridge Port for a virtual switch.

Unless you have special requirements, the predefined controllers should be sufficient to handle the needs of all virtual switches. However, they are intended for use only as virtual switch controllers. They are not recommended for use as traditional TCP/IP stacks.

Use the VSWITCH CONTROLLER statement to define a range of virtual device addresses that are available to control a CP-defined virtual switch's connection to a real LAN segment through an OSA-Express adapter or for a HiperSockets device defined as a Bridge Port.



Operands

ON

Indicates that the z/VM TCP/IP stack is available to control a virtual switch's connection to a real LAN segment through an OSA-Express adapter or for a HiperSockets device defined as a Bridge Port.

OFF

Indicates that the z/VM TCP/IP stack is not available to control a virtual switch's connection to an OSA-Express adapter or HiperSockets Bridge Port device.

device_addr1 device_addr2

Identifies a virtual device address range.

Use *device_addr1 device_addr2* only if you expect a conflict between the real device addresses associated with a virtual switch and virtual device addresses defined by the TCPIP stack.

device_addr1 identifies the start of a virtual device address range. *device_addr2* identifies the end of a virtual device address range. They are hexadecimal device addresses. The range identifies at least three consecutive virtual device addresses to be grouped for the OSA-Express or HiperSockets Bridge Port adapter. Each connection uses three sequential virtual device addresses. The device range defined must allow three devices for each virtual switch. A larger range will be needed if virtual switches are defined with more than one RDEV for dynamic recovery of device failure.

Use * * to specify FAILOVER_ENABLED or FAILOVER_DISABLED without defining a virtual device address range.

FAILOVER_ENABLED

The Control Program (CP) checks timestamps to confirm that the z/VM TCP/IP stack controller is responding to requests to service a controlled device associated with an active virtual switch. If the stack is not responding, CP moves to a backup z/VM TCP/IP stack controller, if one is available.

FAILOVER_ENABLED is the default.

FAILOVER_DISABLED

CP does not perform timestamp checking to make sure that the stack is responding. You must use this option when TRACE OSD or MORETRACE OSD is in effect. Use this setting only if you are doing problem determination that slows the stack enough that it cannot respond to CP controller requests in a timely manner.

Examples

1. To code a typical VSWITCH CONTROLLER statement, specify the following:

```
VSWITCH CONTROLLER ON
```

2. To code a VSWITCH CONTROLLER statement that defines the virtual addresses to be used when devices are attached to the controller, specify the following:

```
VSWITCH CONTROLLER ON 111 119
```

Note: The range is large enough to allow nine devices, meaning the controller can handle three virtual switches, or a single virtual switch with two backup RDEVs defined.

3. To code a VSWITCH CONTROLLER statement with no virtual address range so that timeout checking is not done while performing problem determination (such as tracing) on the stack, specify the following:

```
VSWITCH CONTROLLER ON * * FAILOVER_DISABLED
```

Usage Notes

- The controller functions are disabled by default. To enable the controller, issue VSWITCH CONTROLLER ON. You may then use the OFF operand to dynamically disable the controller.

- Do not specify *device_addr1 device_addr2* on the VSWITCH CONTROLLER statement unless you anticipate a conflict between the real device addresses associated with a virtual switch and the addresses defined for the TCP/IP stack.

If you do not specify a device address range, the Control Program uses the real device address as the virtual device address when a controlled device is attached to the stack controller.

- CP manages the devices used to control a virtual switch's connection to a real LAN segment through an OSA-Express adapter. CP attaches the devices to the z/VM TCP/IP virtual machine. CP also defines a device of type VSWITCH-OSD to the z/VM TCP/IP stack, concatenating *switchnm* with *vdev* and "DEV" to form the device name and *switchnm* with *vdev* and "LINK" to form the link name. These names appear in the TCP/IP query and trace information. Similar processing is used for a virtual switch's connection to a HiperSockets Bridge Port. In this case, CP defines a device of type VSWITCH-HIP, but creates the device name and link name as above.

DEVICE and LINK statements must not be included in the TCP/IP configuration file for these devices.

- A virtual switch's connection to an OSA-Express adapter or HiperSockets Bridge Port is not operational until an eligible z/VM TCP/IP stack is selected to be the controller for the device. CP selects an eligible z/VM TCP/IP stack to be the controller by either:
 - If CONTROLLER *userid1* is specified on the DEFINE VSWITCH command or System Configuration statement, only *userid1* is selected.
 - If CONTROLLER * is specified or allowed to default, CP selects from any eligible z/VM TCP/IP stacks.

A z/VM TCP/IP stack becomes eligible when:

- An IUCV *VSWITCH statement is included in its CP directory entry.
- The TCP/IP VSWITCH CONTROLLER statement is coded, and has defaulted to be ON or is explicitly set to ON in the TCP/IP configuration file or through an OBEYFILE command.
- The stack has completed initialization.
- The stack has virtual device addresses available for CP to attach the control device.

The virtual address range used by CP is specified in the VSWITCH CONTROLLER TCP/IP configuration statement. If no VDEV range is specified, CP uses the virtual device address (*vdev*) that matches the *rdev* address specified on the DEFINE VSWITCH or SET VSWITCH command.

Note:

1. Do not code DEVICE and LINK TCP/IP configuration statements for the device. Do not attach the device to a TCP/IP controller virtual machine. These steps are handled by DEFINE VSWITCH processing when a controller is selected.
2. Do not define a SPECIAL statement or issue DEFINE NIC for the device.

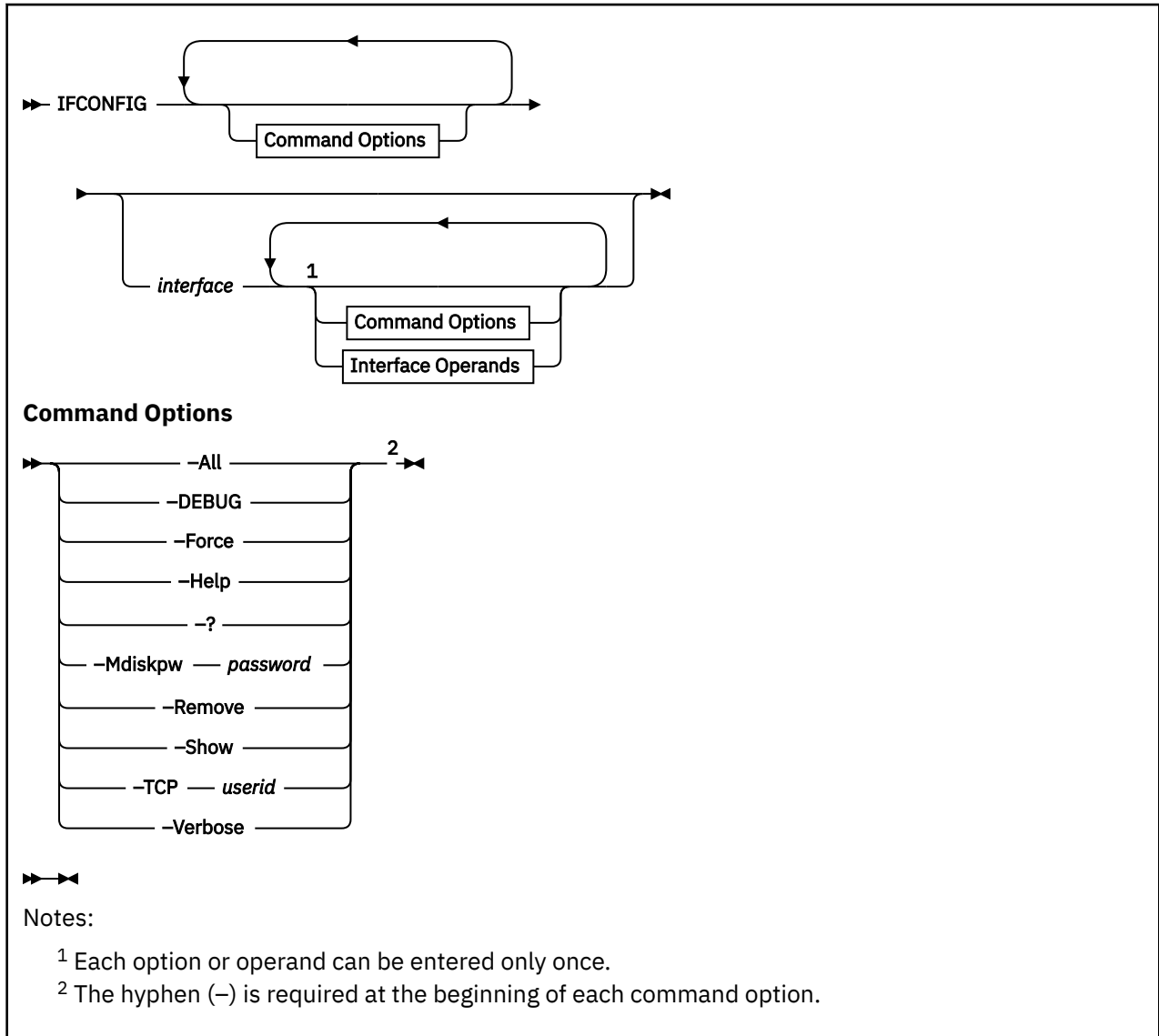
If an eligible stack is not found, you receive a message, and the virtual switch operates in a local LAN environment.

- Use the QUERY CONTROLLER command to find the status of eligible z/VM TCP/IP controller stacks.
- Use the SET VSWITCH command with the DISCONNECT option to stop communication with an OSA-Express adapter. The virtual switch is left in 'Disconnected — operator' mode. For a HiperSockets Bridge Port connection, use the SET VSWITCH BRIDGEPORT DISCONNECT command to stop communication with the HiperSockets device. To restart the device, use the SET VSWITCH BRIDGEPORT CONNECT command.
- When an OBEYFILE VSWITCH CONTROLLER OFF statement is issued for an already active stack, the stack will continue to handle any switches that it is already assigned to control. Likewise, a change to the virtual device address range will not affect any already-defined virtual devices.

Changing the TCP/IP Configuration with the IFCONFIG Command

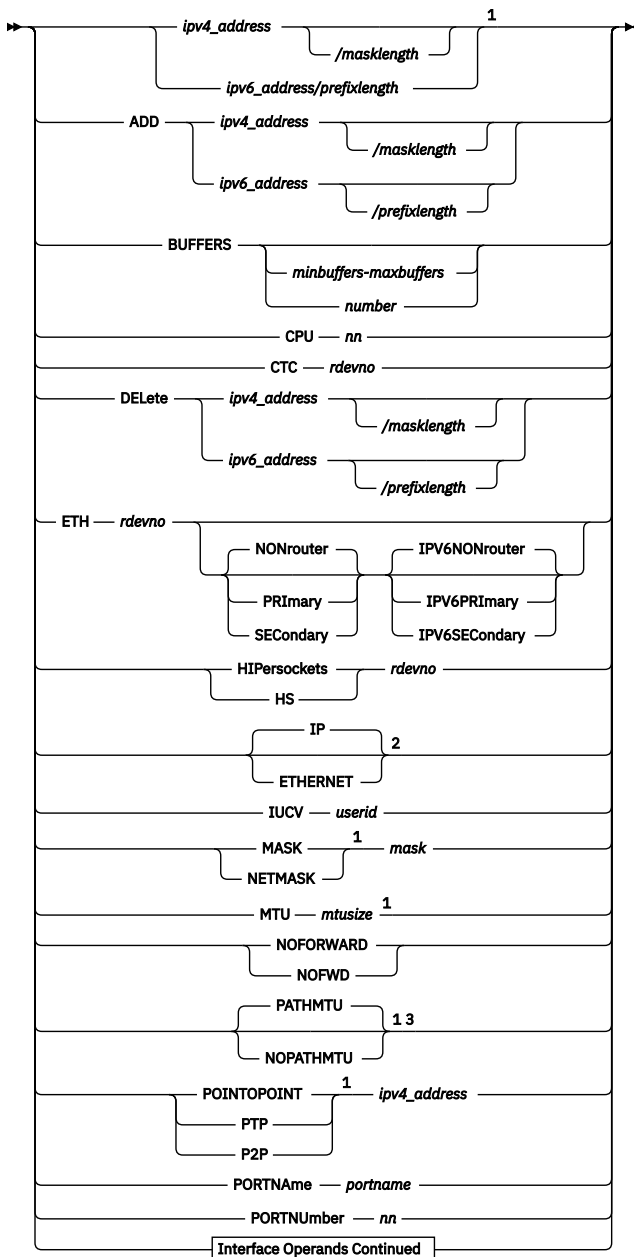
The IFCONFIG command allows you to display information about and make temporary dynamic changes to the TCP/IP configuration without stopping and restarting the TCPIP virtual machine.

IFCONFIG Command



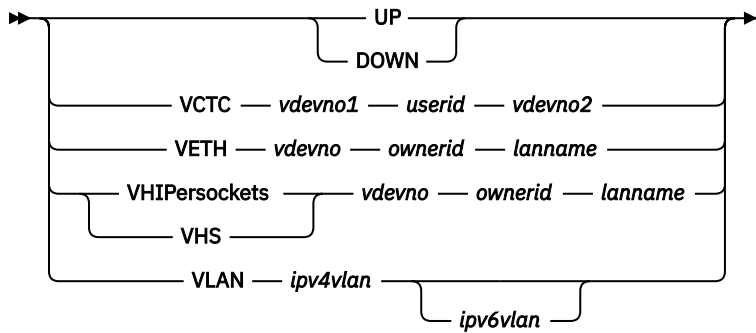


Interface Operands



Notes:

- ¹ This operand can be used to change previously defined devices.
- ² For virtual devices coupling to an existing guest LAN or VSWITCH, the default is the transport type of that guest LAN or VSWITCH.
- ³ PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default.

Interface Operands Continued**Purpose**

Use the IFCONFIG command to configure network interfaces for the z/VM TCP/IP stack or to display the current configuration. It can be used instead of, or in conjunction with, the existing NETSTAT and OBEYFILE commands.

IFCONFIG does not make permanent changes to the network configuration, but it can provide data for this purpose that is compatible with the TCP/IP server configuration file.

Options**-All**

Displays the status of all TCP/IP interfaces.

-DEBUG

Provides information for diagnosing problems associated with the use of this command, in consultation with the IBM TCP/IP support group.

-Force

Bypasses the device consistency verification logic that is used by IFCONFIG when a new device is defined. The device is added to the z/VM TCP/IP configuration.

-Help**-?**

Provides help information about the IFCONFIG command. You cannot specify the HELP parameter on the IFCONFIG command line with other parameters.

-Mdiskpw *password*

Specifies the read password, if required, of the first read/write minidisk on which an obey file generated by IFCONFIG is to reside. IFCONFIG uses the OBEYFILE command to facilitate network interface configuration changes, and in so doing creates a temporary obey file for this purpose.

-Remove

Removes the interface from the configuration. The option detaches a real device or destroys a virtual device. The interface must be inactive before -Remove is specified.

Note: -Remove cannot be specified with any command operands.

-Show

Displays the TCP/IP server configuration file statements that are required to change an existing or define a new interface, but does not change the running system. These statements are device-specific and limited in scope. Before using this information to apply permanent changes to the TCP/IP server configuration, make sure all of the data that is needed for a given statement is provided.

If no interface is specified with the -Show option, information about all known, active interfaces is displayed.

-TCP *userid*

Applies this instance of the IFCONFIG command to a specific z/VM TCP/IP stack virtual machine. The IFCONFIG command communicates with the TCP/IP stack identified by any one of the following:

- The user ID specified with the -TCP option
- The TCPIPID C environment variable (established via the command):

```
GLOBALV SELECT CENV SETL TCPIPID userid
```

- The TcpiUserId value from the TCPIP DATA file
- User "TCPIP"

-Verbose

Displays any CP or NETSTAT commands that are used while changing the running system. To obtain a complete picture of what commands and TCP/IP configuration file statements are required to manually configure z/VM TCP/IP, the -Verbose option can be combined with the -Show option, but then no changes are made to the running system.

Operands

interface

Specifies the name of a network interface assigned to a particular TCP/IP virtual machine. This interface name is limited to 16 characters and cannot begin with a dash (-), end with a colon (:), or contain a semicolon (;).

The interface name corresponds to the link name that is defined on a LINK statement in PROFILE TCPIP.

ipv4_address

The primary IPv4 address of the interface.

masklength

The number of bits used to represent the subnet mask of the interface.

Example: 192.168.0.12/24 indicates that a subnet mask of 255.255.255.0 is to be used with an IPv4 address of 192.168.0.12.

If the *ipv4_address/masklength* format is used rather than simply the *ipv4_address* format, then the NETMASK or MASK operand cannot be used.

Note: The masklength must be between /1 and /30. The IP address must not be the subnet address or broadcast address for the network described by the IPv4 address/masklength pair (that is, it cannot be the first or last address in the subnet's range). For point-to-point links, a masklength of /30 is recommended.

ipv6_address/prefixlength

The primary IPv6 address of the interface, followed by the number of bits used to identify the prefix associated with the address.

Example: fe80:0:0:0:210:a4ff:fee3:956/64 indicates that a prefix of FFFF:FFFF:FFFF:FFFF:0:0:0:0 is to be used with an IP version 6 address of fe80:0:0:0:210:a4ff:fee3:956.

ADD *ipv4_address/masklength*

ADD *ipv6_address/prefixlength*

Adds an IPv4 address and subnet mask pair or IPv6 address and prefix pair to the home list of the interface.

BUFFERS *minbuffers-maxbuffers*

BUFFERS *number*

Specifies a number or a range of input buffers to be allocated for an EQDIO connection. The number of buffers can be between 2048 and 65535. For a range, specify two numbers separated by a single hyphen (-) with no blank spaces — for example: x-y, where x is less than y.

Note: Due to internal device architecture, *minbuffers*, *maxbuffers*, or both values might be rounded to an approximate value. For example, if 3000-65000 is specified, the *minbuffers* and *maxbuffers* values implemented might be 3008-64992. Likewise, a single value such as 50000 might result in a value of 49984 being used.

CPU *nn*

Specifies the virtual processor to be used to run the device driver for the interface. *nn* must be an integer between 0 and 6.

CTC *rdevno*

Defines a real channel-to-channel interface.

DELeTe *ipv4_address/masklength*

DELeTe *ipv6_address/prefixlength*

Removes an IPv4 address and subnet mask pair or IPv6 address and prefix pair from the home list of the interface.

ETH *rdevno routertype*

Defines an EQDIO, LCS, or QDIO Ethernet interface. ETH supports IPv4 and IPv6 router interface types. For IPv4, *routertype* can be PRImary, SECondary, or NONrouter. For IPv6, *routertype* can be IPV6PRImary, IPV6SECondary, or IPV6NONrouter. Only one IPv4 primary router interface (PRImary) can be defined for a QDIO device and only one IPv6 primary router interface (IPV6PRImary) can be defined for a QDIO device. While they can be the same interface, the QDIO devices do not have to be the same interface.

If the router type is specified for an EQDIO or LCS interface, the specification is ignored. OSA Support Facility (OSA/SF) is used to configure the router type for an OSA LCS interface. For more information about OSA/SF, see *Open Systems Adapter-Express Customer's Guide and Reference* (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/iaa2z1f0.pdf).

HIPersockets *rdevno*

HS *rdevno*

Defines a real HiperSockets connection.

IP

ETHERNET

Indicates whether the transport for the link is ETHERNET or IP. An ETHERNET link operates at the Layer 2 level of the OSI model while an IP link operates at Layer 3. This operand can be specified only for real or virtual QDIO Ethernet devices.

Note: Router type is not supported for Layer 2.

IUCV *userid*

Defines an IUCV interface.

MASK *mask*

NETMASK *mask*

Defines or changes the subnet mask that is associated with the interface.

When defining an interface, the default value for the subnet mask is determined by the value of the first octet of the IP address:

If the value of the first octet is:	The default subnet mask is:
0-127	255.0.0.0
128-191	255.255.0.0
192-223	255.255.255.0

The default subnet mask applies only when defining an interface. Unless you use the mask option, changing the IP address of an existing interface leaves the mask unchanged.

Note: The mask must be between 128.0.0.0 and 255.255.255.252. The IP address must not be the subnet address or broadcast address for the network described by the IPv4 address/mask pair (that

is, it cannot be the first or last address in the subnet's range). For point-to-point links, a subnet mask of 255.255.255.252 is recommended.

MTU *mtusize*

Defines or changes the maximum transmission unit (MTU) size that is to be used on the interface. To determine the recommended MTU size, refer to the hardware documentation associated with the device.

If you specify 0 or omit this option, the TCP/IP stack selects an intelligent default. See [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

NOFORWARD
NOFWD

Specifies that packets received on this link are not to be forwarded to another host (that is, packets destined for a foreign host are to be discarded) and that packets transmitted on this link must originate from the local host. Packets received for another host on this link are to be dropped, as are packets received for another host on any link and forwarded through this one. If you do not specify this option, packets received or transmitted on the link can be forwarded to another host.

PATHMTU
NOPATHMTU

Specifies the use of path MTU discovery on IPv4 routes for a given link. PATHMTU is the default when the PATHMTU operand is specified on the ASSORTEDPARMS statement in the TCP/IP configuration file. Otherwise, NOPATHMTU is the default. These operands have no effect on IPv6 routes. Path MTU discovery is always enabled for IPv6 and cannot be disabled.

POINTOPOINT *ipv4_address*
PTP *ipv4_address*
P2P *ipv4_address*

Defines or changes the IP address associated with the other end of a point-to-point interface.

PORTName *portname*

Specifies the Queued Direct I/O (QDIO) port name when it is being defined to be used by this interface. The *portname* is limited to a maximum of 8 characters and is optional. If you specify a port name, all TCP/IP stacks that share a QDIO device must specify the same port name. The port name is optional in all supported levels of the real or virtual OSA-Express adapters.

PORTNumber *nn*

Specifies the physical port or adapter number on the device when it is being defined to be used by this interface. This number depends on the device type:

- For Channel-to-Channel (CTC) connections, specify 0 or 1.
- For IBM Open Systems Adapter-Express operating in LCS mode, specify 0 or 1.
- For IBM Open Systems Adapter-Express operating in QDIO mode, specify a decimal number in the range 0-15. The value of the port number depends on how many ports the OSA-Express adapter supports. If the port number is not specified, it will default to port 0.
- Do not specify a port number for other devices.

UP
DOWN

Starts or stops the interface.

VCTC *vdevno1 userid vdevno2*

Defines a virtual channel-to-channel interface. A virtual CTC is defined and coupled to the specified user's virtual device.

VETH *vdevno ownerid lanname*

Defines a virtual Ethernet connection to the named guest LAN or virtual switch. If EQDIO virtual NICs are supported, an EQDIO VNIC will be defined. Otherwise, a QDIO VNIC will be defined. If a VSWITCH name is specified, *ownerid* must be SYSTEM. If no guest LAN or VSWITCH exists with the specified *ownerid/lanname* combination, a QDIO guest LAN will be created. *ownerid* and *lanname* are limited to a maximum of 8 characters each.

VHipersockets *vdevno ownerid lanname***VHS *vdevno ownerid lanname***

Defines a virtual HiperSockets connection on the named guest LAN. If no guest LAN exists with the specified *ownerid* and *lanname* combination, a HiperSockets guest LAN will be created. The *ownerid* and *lanname* values are limited to a maximum of 8 characters each.

VLAN *ipv4vlan [ipv6vlan]*

Specifies the identifier for a Virtual Local Area Network (VLAN). *ipv4vlan* and *ipv6vlan* are values from 1 to 4094. VLAN can be specified on QDIO Ethernet devices and HiperSockets devices.

For a QDIO Ethernet device, two VLANs can be specified. VLAN *ipv4vlan* specifies the IPv4 VLAN ID. You can optionally specify a separate VLAN ID for your IPv6 network by using *ipv6vlan*. If *ipv6vlan* is not specified, *ipv4vlan* is also used for the IPv6 network.

For a HiperSockets device, only one VLAN ID, *ipv4vlan*, can be specified.

Examples

1. To define an interface to a real HiperSockets adapter, enter the following:

```
ifconfig hsi0 10.1.0.1 mask 255.255.255.240 hs 3904 portname hs5a
```

where:

hsi0 10.1.0.1

is the name and IP address of the HiperSockets interface

mask 255.255.255.240

is the subnet mask associated with the interface

hs 3904

is the real device address of the HiperSockets connection

portname hs5a

specifies the QDIO port name HS5A that is to be used by the interface

Note: A default maximum transmission unit size of 4000 is used.

2. To define an interface to a virtual HiperSockets adapter that is connected to a z/VM guest LAN owned by SYSTEM and named PROD1, enter the following:

```
ifconfig vhsi0 10.2.0.1 vhs 3904 system prod1 mtu 8192
```

where:

vhsi0 10.2.0.1

is the name and IP address of the virtual HiperSockets interface

vhs 3904 system prod1

defines the virtual HiperSockets connection at virtual device number 3904 on LAN name PROD1 of LAN owner SYSTEM.

mtu 8192

specifies that a maximum transmission unit size of 8192 is to be used with this interface

Note: The default subnet mask (255.0.0.0) is used.

3. To define an IUCV interface to user TCPIP2, enter the following:

```
ifconfig iucv0 10.4.0.1/30 pointopoint 10.4.0.2 iucv tcpip2 mtu 9216
```

where:

iucv0 10.4.0.1/30

is the name, IP address, and subnet mask of IUCV interface

pointopoint 10.4.0.2

defines the IP address at the other end of the interface

iucv tcpip2

defines the communication partner user ID as TCPIP2

mtu 9216

specifies that a maximum transmission unit size of 9216 is to be used with this interface

4. To define an interface to a 1Gb Ethernet Open Systems Adapter Express at address 6904-6906 that will participate in a VLAN with an ID of 8, enter the following:

```
ifconfig eth0 192.60.25.6 eth 6904 primary mtu 1500 portname osa69 vlan 8
```

where:

eth0 192.60.25.6

is the name and IP address of the Ethernet Open Systems Adapter Express interface

eth 6904 primary

specifies the QDIO interface with a real device number of 6904 and a router type of PRIMARY

mtu 1500

specifies that a maximum transmission unit size of 1500 is to be used with this interface

portname osa69

specifies the QDIO port name OSA69 that is to be used by the interface

vlan 8

specifies the VLAN ID associated with the interface

Note: The default subnet mask (255.255.255.0) is used.

Now specify the IFCONFIG command with only the interface name, ETH0, to make sure the connection was defined correctly:

```
ifconfig eth0
```

The output will look something like this:

```
ETH0  inet addr: 192.60.25.6 mask: 255.255.255.0
      UP MULTICAST MTU: 1500
      vdev: 6904 rdev: 6904 type: QDIO ETHERNET portname: OSA69
      transport type: IP
      ipv4 router type: PRIMARY ipv6:DISABLED

      cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
      RX bytes: 33932 TX bytes: 15534
```

5. To define an interface to an Ethernet Open Systems Adapter Express (OSA-Express2) at address 3000-3003 that will operate in layer 2 mode, enter the following:

```
ifconfig eth1 10.11.12.26 eth 3000 ethernet mtu 1500
```

where:

eth1 10.11.12.26

is the name and IP address of the Ethernet Open Systems Adapter interface

eth 3000 ethernet

specifies the QDIO interface with a real device number of 3000 and a transport type of ethernet (layer 2)

mtu 1500

specifies that a maximum transmission unit size of 1500 is to be used with this interface

Now specify the IFCONFIG command with only the interface name, ETH1, to make sure the connection was defined correctly:

```
ifconfig eth1
```

The output will look something like this:

```

ETH1    inet addr: 10.11.12.26 mask: 255.0.0.0
        UP BROADCAST MULTICAST MTU: 1500
        vdev: 3000 rdev: 3000 type: QDIO ETHERNET portname: UNASSIGNED
        transport type: ETHERNET MAC address: 02-09-57-00-00-10
        ipv6: DISABLED
        cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
        RX bytes: 110 TX bytes: 0

```

6. To define an interface to IBM Ethernet Open Systems Adapter Express (OSA-Express2) running in LCS mode at address 3612-3613, enter the following:

```
ifconfig lcs0 10.59.12.5 mask 255.255.255.0 ethernet 3612 portnumber 1
```

where:

lcs0 10.59.12.5

is the name and IP address of the Ethernet Open Systems Adapter Express interface

mask 255.255.255.0

is the subnet mask associated with the interface

ethernet 3612

specifies the LCS interface with a real device number of 3612

portnumber 1

specifies that port number 1 on the OSA adapter is to be used by this interface

Note: A default maximum transmission unit size of 1500 is used.

Now specify the IFCONFIG command with only the interface name, LCS0, to make sure the connection was defined correctly:

```
ifconfig lcs0
```

The output will look something like this:

```

LCS0    inet addr: 10.59.12.5 mask: 255.255.255.0
        UP BROADCAST MULTICAST MTU: 1500
        vdev: 3612 rdev: 3612 type: LCS ETHERNET portnumber: 1
        cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
        RX bytes: 13965 TX bytes: 38904

```

7. To display information about an interface that is not up, enter the following:

```
ifconfig lcs1
```

The output will look something like this:

```

LCS1    inet addr: 10.200.13.78 mask: ?
        DOWN BROADCAST MULTICAST MTU: ?
        vdev: 1F08 rdev: 1F08 type: LCS TETHERNET portnumber: 0
        Broadcast type: Local  Arp MAC Address: Canonical
        cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
        RX bytes: 0 TX bytes: 0

```

Note: Question marks (?) are displayed for the subnet mask and the maximum transmission unit size because they are not known.

8. To display the TCP/IP configuration file statements needed to define a channel-to-channel (CTC) interface, enter the following:

```
ifconfig ctc0 10.14.6.1/30 vctc 800 tcpip2 3600 ptp 10.14.6.2 portnumber 0 -show
```

where:

ctc0 10.14.6.1/30

is the name, IP address, and subnet mask of the CTC interface

vctc 800 tcpip2 3600

is the virtual CTC at virtual device number 800 that is coupled to virtual device number 3600 owned by user TCPIP2

ptp 10.14.6.2

is the IP address at the other end of the interface

-show

displays the TCP/IP server configuration file changes

Note: Because an MTU was not specified on the command, the LINK MTU is configured as 0, forcing the stack to choose an intelligent default according to [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

The output will look something like this:

```
; Generated by <IFCONFIG ctc0 10.14.6.1 vctc 800 tcpip2 3600
; ptp 10.14.6.2 portnumber 0 -show>
; 25 Mar 2025 12:16:57
DEVICE DEV@800 CTC 800
LINK CTC0 CTC 0 DEV@800 MTU 0
HOME
10.14.6.1 255.255.252 CTC0
START DEV@800
```

9. To display the TCP/IP configuration file statements necessary to define a virtual HiperSockets interface to another virtual HiperSockets connection, enter the following:

```
ifconfig vhs3 10.2.0.1 mask 255.255.0.0 vhs 3500 tcpip41 subnt96 -show
```

where:

vhs3 10.2.0.1

is the name and IP address of the virtual HiperSockets interface.

mask 255.255.0.0

is the subnet mask associated with the interface.

vhs 3500 tcpip41 subnt96

defines the virtual HiperSockets connection at virtual device number 3500 on LAN name SUBNT96 owned by user TCPIP41.

-show

displays the TCP/IP server configuration file changes.

Note: Because an MTU was not specified on the command, the LINK MTU is configured as 0, forcing the stack to choose an intelligent default according to [“Intelligent Default MTU Values Based on the Device and Link Type”](#) on page 539.

The output will look something like this:

```
; Generated by <IFCONFIG vhs3 10.2.0.1 mask 255.255.0.0 vhs
; 3500 tcpip41 subnt96 -show>
; 25 Mar 2025 12:20:54
DEVICE DEV@3500 HIPERS 3500
LINK VHS3 QDIOIP DEV@3500 MTU 0
HOME
10.2.0.1 255.255.0.0 VHS3
START DEV@3500
```

10. To connect a virtual EQDIO device (if supported) to an existing QDIO guest LAN named TEST1 that is owned by user TCPIP41, display any CP or NETSTAT commands that were used to change the system, and apply these changes to a specific TCP/IP stack virtual machine named TCPIP39, enter the following:

```
ifconfig eth0 10.1.1.1 veth 3400 tcpip41 test1 -v -tcp tcpip39
```

where:

eth0 10.1.1.1

is the name and IP address of the virtual EQDIO Ethernet interface (if supported)

veth 3400 tcpip41 test1

is the existing EQDIO connection (if supported) at virtual device number 3400 named TEST1 that is owned by user TCPIP41

-v

specifies the -Verbose command option, which displays any CP or NETSTAT commands that were used to change the running system

-tcp tcpip39

applies these changes to a z/VM TCP/IP stack virtual machine named TCPIP39

The output will look something like this:

```
* NETSTAT TCP TCPIP39 CP QUERY LAN TEST1 OWNER TCPIP41
* NETSTAT TCP TCPIP39 CP DEFINE NIC 3400 EQDIO
* NETSTAT TCP TCPIP39 CP COUPLE 3400 TCPIP41 SUBNT176
* OBEYFILE IFCONFIG CMSUT1 A ( TCP TCPIP39
```

11. To define a real HiperSockets connection, display any CP or NETSTAT commands that were used to change the system, and apply these changes to a specific TCP/IP stack virtual machine named TCPIP2, enter the following:

```
ifconfig hs0 10.1.2.1 hs fb00 -v -tcp tcpip2
```

where:

hs0 10.1.2.1

is the name and IP address of the HiperSockets interface

hs fb00

is the real device address of the HiperSockets connection

-v

specifies the -Verbose command option, which displays any CP or NETSTAT commands that were used to change the running system

-tcp tcpip2

applies these changes to a z/VM TCP/IP stack virtual machine named TCPIP2

The output will look something like this:

```
* NETSTAT TCP TCPIP2 CP ATTACH FB00 TCPIP2 FB00
* NETSTAT TCP TCPIP2 CP ATTACH FB01 TCPIP2 FB01
* NETSTAT TCP TCPIP2 CP ATTACH FB02 TCPIP2 FB02
* OBEYFILE IFCONFIG CMSUT1 A ( TCP TCPIP2
```

12. To define a virtual CTC connection and display any CP or NETSTAT commands that were used to change the system, enter the following:

```
ifconfig ctc0 10.1.3.1/30 vctc 800 tcpip2 3600 ptp 10.1.3.2 portnum 1 -v
```

where:

ctc0 10.1.3.1/30

is the name, IP address, and subnet mask of the CTC interface

vctc 800 tcpip2 3600

is the virtual CTC at virtual device number 800 that is coupled to TCPIP2 user's virtual device number 3600

ptp 10.1.3.2

defines the IP address at the other end of the interface

portnum 1

specifies the adapter number of the CTC device

-v

specifies the -Verbose command option, which displays any CP or NETSTAT commands that were used to change the running system

The output will look something like this:

```
* NETSTAT TCP TCPIP CP DEFINE 3088 800
* NETSTAT TCP TCPIP CP DEFINE 3088 801
* NETSTAT TCP TCPIP CP COUPLE 800 TCPIP2 3600
* NETSTAT TCP TCPIP CP COUPLE 801 TCPIP2 3601
* OBEYFILE IFCONFIG CMSUT1 A ( TCP TCPIP
```

- To define an interface to a real Open Systems Adapter Express at address 5200-5202 using IP version 6, enter the following:

```
ifconfig eth0 1080:0:0:0:210:a4ff:fee3:956/64 eth 5200 secondary ipv6primary portname osa52
```

where:

eth0 1080:0:0:0:210:a4ff:fee3:956/64

is the name, IPv6 address and prefix length of the Ethernet Open Systems Adapter Express interface

eth 5200 secondary ipv6primary

specifies the QDIO interface with a real device number of 5200 as a secondary IPv4 router and a primary IPv6 router

portname osa52

specifies the QDIO port name OSA52 that is to be used by the interface

Note: A default maximum transmission unit size of 1500 is used.

- To add the IPv4 address 192.168.0.9 with a subnet mask of 255.255.255.0 to interface eth1 (which must be defined already), enter the following:

```
ifconfig eth1 add 192.168.0.9/24
```

where:

eth1

is the interface name

add 192.168.0.9/24

is the IPv4 address to be added to the interface followed by the number of high order bits specified by the subnet mask for the interface

- To delete the IPv6 address 1080:0:0:0:AB32:800:FF83:10 with prefix length of 64 from interface eth3, enter the following:

```
ifconfig eth3 delete 1080:0:0:0:AB32:800:FF83:10/64
```

where:

eth3

is the interface name

delete 1080:0:0:0:AB32:800:FF83:10/64

is the IPv6 address to be deleted from the interface followed by the prefix length associated with that address

- In this example, *eth0* and *eth1* are attached to the 9.60.59.0/26 subnet. The following shows the IFCONFIG output after *eth1* has experienced an outage:

```
ifconfig -all
ETH0      inet addr: 9.60.59.6 mask: 255.255.255.192
          UP BROADCAST MULTICAST MTU: 1500
          vdev: 0600 type: QDIO ETHERNET portname: UNASSIGNED
          transport type: IP
          ipv4 router type: NONROUTER ipv6: DISABLED
          LAN owner: SYSTEM name: SUBNTA
          cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
```

```

RX bytes: 0 TX bytes: 0

ETH1    inet addr: 9.60.59.7 mask: 255.255.255.192
        DOWN MTU: 1500
        vdev: 0700 type: QDIO ETHERNET portname: UNASSIGNED
        transport type: IP
        ipv4 router type: NONROUTER ipv6: DISABLED
        LAN owner: SYSTEM name: SUBNTA
        cpu: 0 forwarding: ENABLED ipv4 path MTU discovery: DISABLED
        RX bytes: 0 TX bytes: 0
        IPv4 Takeover Link: ETH0

```

The IPv4 Takeover Link field in the *eth1* output shows that IPv4 ARP responsibility for *eth1*'s IP addresses has been taken over by *eth0*. For more information about interface takeover, see [“Interface Takeover for Local Area Networks” on page 513](#).

Usage Notes

- Entering the IFCONFIG command with no other parameters specified displays all active interfaces.
- Use the IFCONFIG command to configure the following types of interfaces:
 - CTC
 - EQDIO
 - HiperSockets
 - IUCV
 - LCS
 - QDIO
- After an interface is defined using IFCONFIG, the only operands that can be changed are the IP address, subnet mask, peer IP address and MTU size. The interface can also be started and stopped. A particular real device can be defined only once to TCP/IP.
- There are certain LCS devices that have multiple network adapters, such as the IBM Open Systems Adapter. IFCONFIG does not permit a second interface to be defined that references a device number already in use by TCP/IP. Therefore, an additional device must be defined for each additional network adapter that you wish to use.
- While IFCONFIG attempts to make sure that the chosen device number and interface type are compatible, there are circumstances where LCS and CTC devices are indistinguishable until TCP/IP actually starts using the device. In these cases the file *StackID* TCPERROR, which contains details of the error, is sent to the issuer of the IFCONFIG command. Depending on the exact nature of the error, it might be necessary to recycle the z/VM TCP/IP stack in order to delete the erroneous device from the TCP/IP configuration.
- In order to use IFCONFIG to create a device that is enabled for IPv6, you must specify an IPv6 address either as the primary address or on the ADD operand of the initial command that is used to create the device. Otherwise, even though the device itself is IPv6 capable, it is not enabled in the TCP/IP stack and cannot be enabled later without recycling the stack.
- VLAN configuration information can be found in [Planning for Guest LANs and Virtual Switches in z/VM: Connectivity](#).
- Be aware that the IFCONFIG command makes use of the NETSTAT and OBEYFILE commands to facilitate its operations. To use this command to make network interface changes, your user ID must be included in the OBEY list in the TCP/IP server configuration file.

Note also that one or more considerations regarding use of the NETSTAT and OBEYFILE commands might apply when you use the IFCONFIG command. For specific information, see the Usage Notes for these commands.

- When running MPRoute, various defaults are taken when dealing with interfaces not defined in the MPRoute configuration file. This could cause undesirable routing scenarios when using IFCONFIG to configure new interfaces to the stack. In order to avoid this, make sure you have:

OBEYFILE Command

```
GLOBAL_OPTIONS
IGNORE_UNDEFINED_INTERFACE=YES;
```

specified in the MPRoute configuration file if you plan on using IFCONFIG in this manner.

- The amount of information returned on an IFCONFIG query could differ depending on whether or not the user issuing the command is in the OBEY list of the TCP/IP server.

Return Codes

Return Code

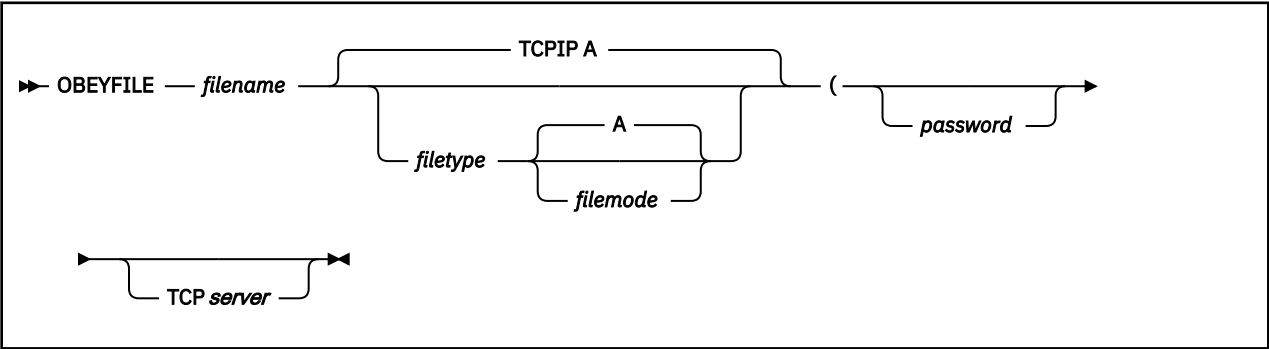
Description

0	The command completed successfully.
4	The command completed successfully, but a warning condition was detected.
8	The command was not specified correctly.
12	An error was encountered.
16	An unexpected condition was encountered.

Changing the TCP/IP Configuration with the OBEYFILE Command

The OBEYFILE command allows you to make temporary dynamic changes to the system operation and network configuration without stopping and restarting the TCPIP virtual machine. You can maintain different files that contain a subset of the TCP/IP configuration statements and use OBEYFILE to activate them while TCP/IP is running.

OBEYFILE Command



Purpose

Use the OBEYFILE command to redefine your TCP/IP configuration without interrupting it. These changes are temporary and may be altered if another OBEYFILE command is issued or when TCP/IP is restarted.

The OBEYFILE command is issued from CMS. It instructs TCP/IP to read a new configuration information file (the *obey file*) while it is running. In order to issue this command, your user ID must be included in the OBEY list in the configuration file. When an unauthorized user issues an OBEYFILE command, audit messages are written to the TCPIP virtual machine console.

Some statements are ignored during OBEYFILE processing and others have limitations. These restrictions are summarized in the Usage Notes section below.

Operands

filename

filetype

filemode

The name of a CMS file that contains TCP/IP configuration statements. The file type defaults to TCPIP and the file mode defaults to A.

password

The CP minidisk read password. If the password is "ALL" or if the minidisk is protected by an external security manager such as RACF, then omit the password.

TCP *server*

The user identifier of the TCP/IP virtual machine whose configuration you want to change. If this option is not used, OBEYFILE addresses the TCP/IP machine identified in the TCPIP DATA file.

Examples

- In this example, file TRACE TCPIP A contains configuration statements to activate ping tracing and direct trace output to file TCPIP LOGFILE L. The user minidisk containing file TRACE TCPIP is accessed as file mode A and is protected by RACF. The TCPIP virtual machine has been given READ access to this minidisk.

```
FILE TCPIP LOGFILE L
TRACE PING
```

The trace is enabled by the command:

```
obeyfile trace
```

Enable the trace for the TCPTTEST virtual machine using the command:

```
obeyfile trace (tcp tcptest
```

- In this example, NOTRACE TCPIP B contains statements to turn off all tracing, close the current trace file, and route any future trace output to the console. The CP read password for the minidisk accessed as B is "readpw".

```
NOTRACE
SCREEN
```

The trace is disabled by the command:

```
obeyfile notrace tcpip b (readpw
```

Usage Notes

- The obey file must reside on a minidisk. The OBEYFILE command does not support the CMS Shared File System (SFS).
- All files on the minidisk where the designated obey file resides are closed when that file is processed by the OBEYFILE command.
- TCP/IP ignores the CONNECTEXIT, PORT, TN3270EEXIT, and TRANSFORM parameters on any INTERNALCLIENTPARMS statement included in the obey file.
- TCP/IP ignores any MONITORRECORDS statements included in the obey file.
- You cannot add new DEVICE and LINK statements for Offload devices using OBEYFILE, nor can you modify or delete any existing DEVICE or LINK statement.
- When you add or change a configuration statement using OBEYFILE, be aware that the existing statement is replaced. Therefore, the obey file must include the *entire* statement, not just the new or changed portions.

For example, when you add new LINK statements, all the entries are deleted from the GATEWAY, HOME, and TRANSLATE statements. Be sure to include the complete entries for the GATEWAY, HOME, and TRANSLATE statements when adding new LINK statements in the obey file.

- The following statements cause an error and should not be included in an obey file:

ACBPOOLSIZE	NCBPOOLSIZE
ADDRESSTRANSATIONPOOLSIZE	RCBPOOLSIZE
CCBPOOLSIZE	SCBPOOLSIZE
DATABUFFERPOOLSIZE	SKCBPOOLSIZE
ENVELOPEPOOLSIZE	SMALLDATABUFFERPOOLSIZE
FIXEDPAGESTORAGEPOOL	TCBPOOLSIZE
FOREIGNIPPOOLSIZE	TINYDATABUFFERPOOLSIZE
IPROUTEPOOLSIZE	UCBPOOLSIZE
LARGEENVELOPEPOOLSIZE	

- A successful OBEYFILE command is confirmed by the message:

```
StackID has read and obeyed file filename filetype
```

- When there is a problem with OBEYFILE, file *StackID* TCPERROR is returned containing a description of the error and an error message is displayed:

```
StackID says: Configuration error. Details are in StackID TCPERROR.
```

- Some functions of OBEYFILE may be accomplished using the OBEY function of the NETSTAT command. Refer to the *z/VM: TCP/IP User's Guide* for information about this command.
- If you add new DEVICE and LINK statements using OBEYFILE, the SMTP server must be recycled in order to recognize them.

Return Codes

Return Code	Description

0

The command completed successfully.

4

The OBEYFILE command was successful; however, the stack found problems with the input file. Details are the file *StackID* TCPERROR.

8

The stack could not read the input file.

12

The command contains a syntax error in the input file.

16

The TCP/IP stack is not available for use.

20

A fatal error has occurred while processing the OBEYFILE command.

24

The user is not authorized to issue the command.

Starting and Stopping TCP/IP Services

Specific TCP/IP services that are managed by server virtual machines other than the TCP/IP stack can also be started and stopped (and in some cases, modified while a server continues operation) on a server-specific basis. For more information, see [“Server Administrative Command Interface Summary” on page 51](#).

In order to control the availability of TCP/IP services, there are procedures to start and stop the TCPIP virtual machine (the stack) and its related service virtual machines. See [“Starting TCP/IP Servers”](#) on page 52 for more information.

Note: You cannot logon the TCP/IP server via a Telnet connection that is managed by that same server. For more information, see [“TCP/IP and SSL Server Logon Restrictions”](#) on page 52.

Chapter 17. Configuring the UFT Server

This chapter describes how to configure the Universal File Transfer (UFT) server (daemon). To configure the UFT server virtual machine, you must perform the following steps:

UFT Server Configuration Steps
1. Update the TCPIP server configuration file. 2. Update the DTCPARMS file for the UFT server. 3. Update the TCPIP DATA file. 4. Customize the UFTD CONFIG file 5. Perform advanced UFT server configuration, if needed.

Dynamic System Operation: The UFT server provides a console subcommand interface that allows you to perform various server administration tasks. For more information see [“Dynamic Server Operation”](#) on page 656.

Step 1: Update PROFILE TCPIP

Include the UFT server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The UFT server is then automatically started when TCPIP is initialized. The IBM default user ID for this server is **UFTD**. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
  UFTD      0
```

The UFT server requires port TCP 608 to be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  608  TCP UFTD      ; UFTD Server
```

Step 2: Update the DTCPARMS File

When the UFT server is started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to this server. Tags that affect the UFT server are:

```
:nick.UFTD
:Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33.

UFTD Command

```
➤ UFTD ➤
```

Purpose

UFT services are initiated using the UFTD command.

Operands

The UFTD command has no operands.

Step 3: Update the TCPIP DATA File

To allow CMS users to be aware of the true origin of files received by the UFT server, add the following statement to the TCPIP DATA file:

```
UFTSERVERID UFTD
```

Step 4: Customize the UFTD CONFIG File

The UFT server uses one configuration file, UFTD CONFIG. This file is used to specify operational parameters for the UFT server virtual machine, such as:

- DBCS conversion options
- Identification of clients
- Maximum size of files received
- Name resolution of connecting clients
- Port that UFT listens on
- Protocol command user exits.

See “[UFT Configuration File Statements](#)” on page 650 for detailed information about how to specify entries within this file. A sample UFT configuration file is provided as UFTD SCONFIG on the TCPMAINT 591 disk. Your customized UFT configuration file should be copied to the TCPMAINT 198 minidisk as UFTD CONFIG.

UFT Configuration File Statements

Specify UFT server parameters in the UFT configuration file as described in this section. Keep in mind the following when configuration statements are specified:

- Tokens are delimited by blanks and record boundaries.
- All characters to the right of, and including, a semicolon are treated as comments.

Note: GLOBALV values used by the UFT server are defined in the GLOBALV group **UFTD**. GLOBALV values override coded defaults, and are overridden by configuration file statements.

IDENTIFY Statement

The IDENTIFY statement specifies that the UFT server should attempt to identify the client by the "identify protocol" described by RFC 1413. If an IDENT server is available at the remote host the client is connecting from, the identity of that client (that is, a user ID) is obtained and saved.



Operands

YES

Indicates that an attempt should be made to connect to the remote host's IDENT server and obtain the identity of the client. If the remote site has an IDENT server, the returned information is saved in the GLOBALV CLIENT variable. If TRACE ALL or TRACE IDENTIFY is active, the results are added to the trace output as well. If no IDENTIFY statement is specified in the configuration file, IDENTIFY YES is assumed.

NO

Indicates that no request should be sent to the remote host's IDENT server.

MAXFILEBYTES Statement

The MAXFILEBYTES statement specifies the maximum size, in bytes, of mail that is accepted over a TCP connection.



Operands

*

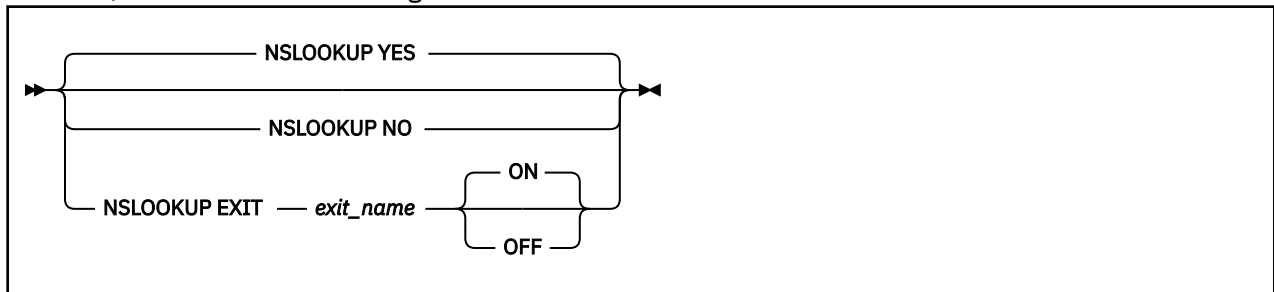
The default byte size limit. This has the special meaning of no limit.

bytes

The maximum number of bytes to accept. Files arriving that are larger than this size are rejected. The limits for this statement are 1 and 2,147,483,647.

NSLOOKUP Statement

The NSLOOKUP statement specifies that resolution of the client connection address should be done. Once resolved, the results are saved in global variables and traced if TRACE ALL or TRACE NSLOOKUP is on.



Operands

YES

Indicates that a DNS lookup should be performed against the client IP address. The resulting host name and IP address are saved in the GLOBALV variables HOSTNAME and HOSTADDR. If TRACE ALL or TRACE NSLOOKUP is active, the results are added to the trace output as well. If NSLOOKUP is not specified in the UFT configuration file, NSLOOKUP YES is assumed.

NO

Indicates that no DNS lookup of the client IP address should be performed.

EXIT *exit_name*

Indicates that a system administrator-provided exit exec should be called to determine what client IP address verification should be done. Based on the return code from the exec, the connection can be verified, not verified, or rejected. See “DNS Lookup Exit” on page 655 for details on the interface to this exit. The exec name defaults to **UFTNSLKX**.

ON

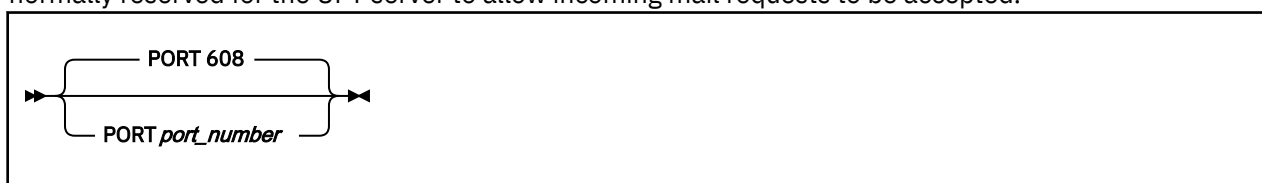
Indicates the exit exec should be invoked for all client resolution activity. This is the default.

OFF

Indicates the exit function should be initialized, but not yet invoked, for client resolution. A NSLOOKUP EXIT ON subcommand is required to initiate exit processing of client resolution.

PORT Statement

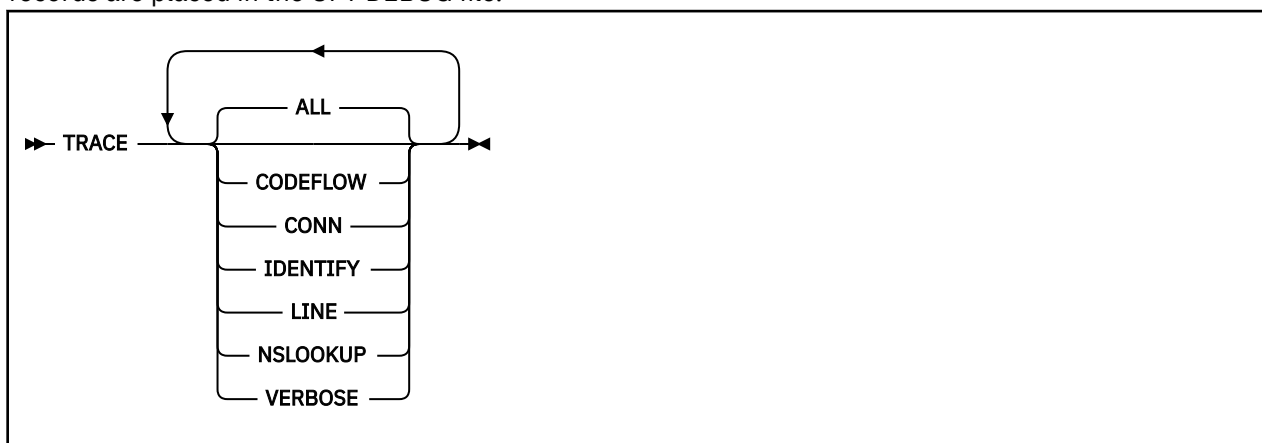
The PORT statement causes the UFT server to listen on a specific port. By convention, port number 608 is normally reserved for the UFT server to allow incoming mail requests to be accepted.

**Operands*****port_number***

An integer in the range of 1 through 65,535 that specifies the port number to which the UFT server listens. The default is port 608.

TRACE Statement

The TRACE statement specifies which type of tracing should be turned on during initialization. Trace records are placed in the UFT DEBUG file.

**Operands****ALL**

Initiates tracing of all types.

CODEFLOW

Initiates tracing of UFT code flow.

CONN

Initiates tracing of connectivity activity.

LINE

Initiates tracing of all commands and replies and their associated connection number.

IDENTIFY

Initiates tracing of the IDENTIFY function and results.

NSLOOKUP

Initiates tracing of the resolver function and results.

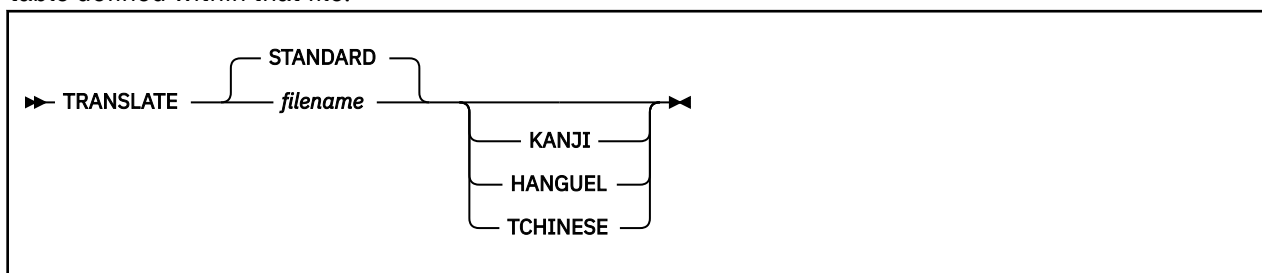
VERBOSE

Initiates tracing of a collection of UFT data, equivalent to specifying CODEFLOW, CONN and LINE.

If no trace statement is found in the UFT configuration file, no tracing is initiated. Multiple TRACE statements may be used to start more than one trace activity.

TRANSLATE Statement

The TRANSLATE statement specifies the file name of a SBCS or DBCS translation table to be used by the UFT server. When a translation file is specified, received ASCII file data is converted using the requested table defined within that file.



Operands

filename

Specifies the file name of the translation table to be used. The default file name is STANDARD and the file type defaults to TCPXLBIN unless a DBCS option (KANJI, HANGEUL or TCHINESE) is specified.

KANJI

Specifies the data received will contain DBCS strings and that a Kanji DBCS translation table should be used. The file type of the binary translation file must be TCPKJBIN.

HANGUEL

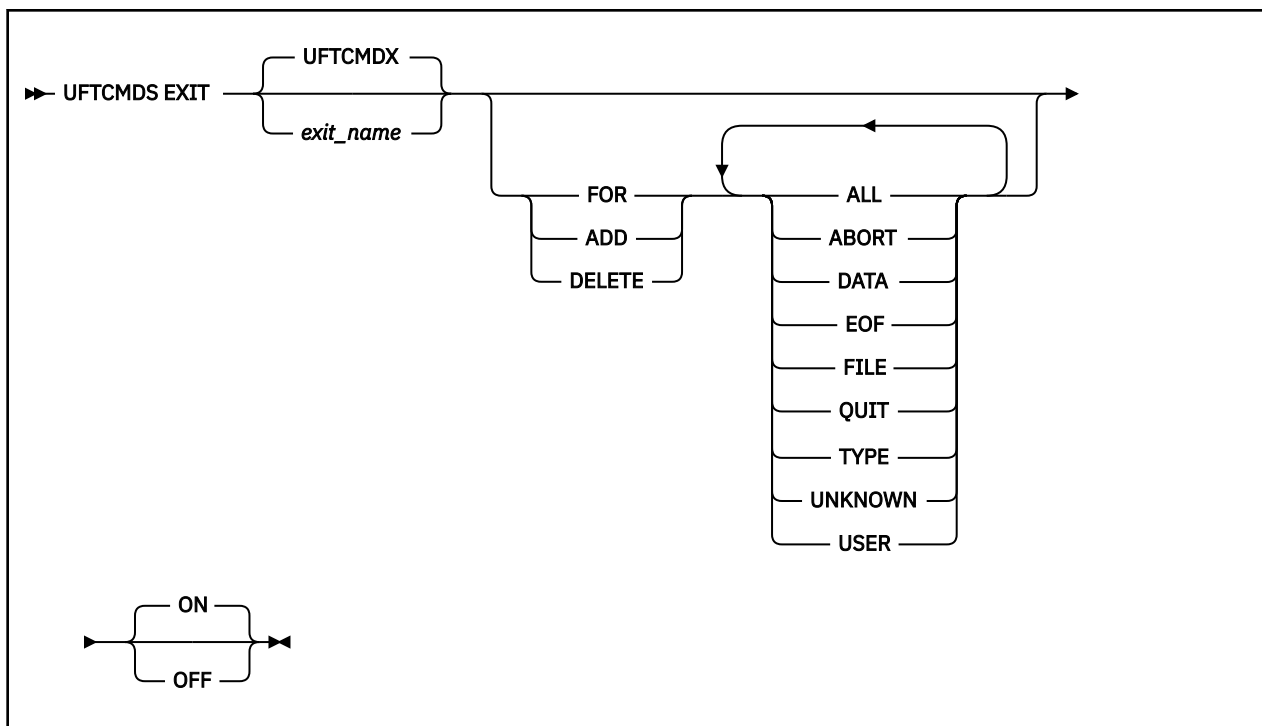
Specifies the data received will contain DBCS strings and that the Hangeul DBCS translation table should be used. The file type of the binary translation file must be TCPHGBIN.

TCHINESE

Specifies the data received will contain DBCS strings and that the Traditional Chinese DBCS translation table should be used. The file type of the binary translation file must be TCPCHBIN.

UFTCMDS EXIT Statement

The UFTCMD5 EXIT statement specifies which UFT protocol commands should be processed by an administrator-provided exit exec.



Operands

exit_name

Name of the exec to be called to handle any initiated protocol command exits. This exec is called on receipt of any commands for which the exit is enabled. See “Protocol Commands Exit” on page 655 for details on the exit interface. The default exec name is **UFTCMDX**.

FOR

Defines which UFT protocol commands are enabled for the exit. If the UFTCMDS exit was previously defined, any commands specified with FOR replace all previously enabled commands.

ADD

Defines which UFT protocol commands are to be added to any existing commands. The enabled commands will be the sum of any previously enabled commands and the commands specified with ADD.

DELETE

Defines which UFT protocol commands are to be deleted from exit processing. The exit remains active for any remaining enabled commands, if any.

command name

The *command name*, which consists of ALL, ABORT, DATA, EOF, FILE, QUIT, TYPE, UNKNOWN, and USER, identifies which protocol commands cause the exit to be called. For any individual command, any valid representation of the command or its synonyms, causes the exit to be called. Two keywords are not commands, but have special meaning. ALL indicates the exec is to be called for all UFT commands eligible for exit processing, and UNKNOWN indicates the exit is to be called for any unknown commands received.

ON

Indicates the exit exec should be invoked for any enabled UFT commands. This is the default.

OFF

Indicates the exit function should be initialized, but not yet invoked, for any UFT protocol commands. A UFTCMDS EXIT ON command is then required to initiate exit processing of UFT protocol commands.

Step 5: Advanced Configuration Considerations

The server exits described in this section can provide greater control over how files are processed by the UFT server. These exits might be used to:

- reject files from a particular host, based on an IP address.
- reject particular UFT protocol commands or control file processing performed by the UFT server, based on available "state" information.

Prior to customizing the server exits described in this section, ensure that you have reviewed the exit limitations and customization recommendations presented in [“Customizing Server-specific Exits”](#) on page 49.

DNS Lookup Exit

The DNS lookup exit, when enabled, is called for each new UFT client connection made to the server.

DNS Lookup Exit Input

The following blank delimited arguments are passed to the exit:

Argument	Value	Format
1	Version of plist (1 currently)	Integer
2	Port numbers of UFT Server	Integer
3	IP Address of UFT Server	Character
4	Port number of the Client or 0 if not available	Integer
5	Source IP address of Client	Character
6	User value 1 (null)	Any

Return Codes

The UFT server recognizes one of the following exit return codes:

Return Code	Meaning
0	Accept client as valid (no verification)
1	Perform verification (same as NSLOOKUP ON)
2	Close connection
3	Disable the exit function

A sample UFT configuration file is provided as UFTNSLKX SAMPEXEC on the TCPMAINT 591 disk. Your customized UFT configuration file should be copied to the TCPMAINT 198 minidisk as UFTNSLKX EXEC. For more information about the supplied DNS lookup exit, refer to the content of the supplied **UFTNSLKX SAMPEXEC** file.

Protocol Commands Exit

The protocol commands exit, when enabled for a given command, is called once for each UFT protocol command received by the server.

Argument	Value	Format
1	Version of plist (1 currently)	Integer
2	Port number of UFTD Server	Integer

Argument	Value	Format
3	IP address of the UFTD Server	Character
4	Port number of the client or 0 if not available	Integer
5	Source IP address of client	Character
6	Client host domain name, (if resolved) or keyword "UNKNOWN"	Character
7	Command being processed	Character
8	Length of command	Integer
9	Remainder of command buffer enclosed in single quotation marks	Character
10	Length of buffer	Integer
11	User value 1	Any
12	User value 2	Any

Note: User values 1 and 2 are initially null for the first call to the exit. Returned values are saved and passed as input to subsequent calls to the exit for the life of the connection. Once the connection is closed, the user values are reset to null for the next call on a new client connection.

Return Code	Meaning
0	Continue processing command
1	Reject this command as out of sequence (503)
2	Close the connection, abort transfer (526)
3	Return the response code contained in User value 1 to the client. The response code must exist in the UFT message repository (UFTUME) as a protocol response message.
4	Disable the exit function

A sample UFT configuration file is provided as UFTCMDX SAMPEXEC on the TCPMAINT 591 disk. Your customized UFT configuration file should be copied to the TCPMAINT 198 minidisk as UFTCMDX EXEC. For more information about the supplied protocol commands exit, refer to the content of the supplied **UFTCMDX SAMPEXEC** file.

Protocol Commands Exit Input

The following blank delimited arguments are passed to the exit:

Return Codes

The UFT server recognizes the following exit return codes:

Dynamic Server Operation

Some configuration attributes (such as tracing and exit enabling) can be changed during server operation by using the UFTD subcommands described in the next section. In addition, certain server activities can be queried or changed by subcommands. Any subcommand not understood by the UFT server is assumed to be a valid CMS command and is passed to the CMS command line for execution.

Issue UFTD subcommands at the UFT server console.

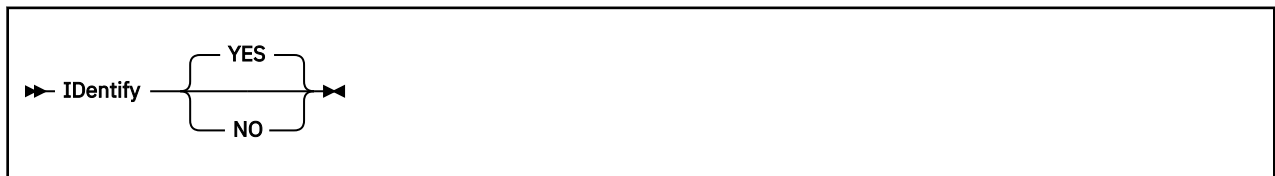
UFTD Subcommands

The UFTD subcommands are listed in [Table 47 on page 657](#). This table provides the shortest abbreviation, a description, and a reference for more information for each UFTD subcommand.

Table 47. UFTD Subcommands

Subcommand	Minimum Abbreviation	Description	Location
IDENTIFY	ID	Causes the server to obtain the identity of a client.	“IDENTIFY Subcommand” on page 657
NSLOOKUP	NSL	Causes the server to perform a DNS lookup.	“NSLOOKUP Subcommand” on page 658
QUERY	Q	Displays configuration, exit and tracing status.	“QUERY Subcommand” on page 658
QUIT	QUIT	Terminates the server.	“QUIT Subcommand” on page 659
STOP	STOP	Terminates the server.	“STOP Subcommand” on page 659
TRACE	TR	Activates or deactivates server tracing.	“TRACE Subcommand” on page 660
UFTCMDS EXIT	UFTCMDS EXIT	Adds or removes protocol command exit points on the running system.	“UFTCMDS EXIT Subcommand” on page 661

IDENTIFY Subcommand



Purpose

The IDENTIFY subcommand causes the server to obtain the identity of a client.

Operands

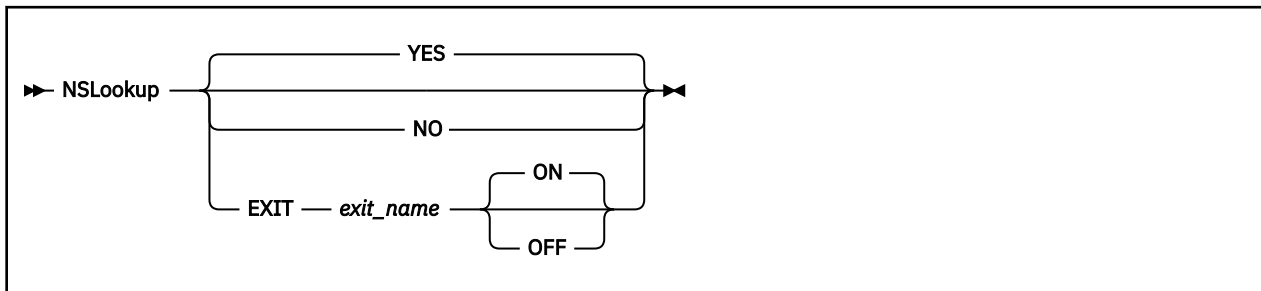
YES

Indicates that an attempt should be made to connect to the remote host's IDENT server and obtain the identity of the client. If the remote site has an IDENT server, the returned information is saved in the GLOBALV CLIENT variable. If TRACE ALL or TRACE IDENTIFY is active, the results are added to the trace output as well. This is the default.

NO

Indicates that no request should be sent to the remote host's IDENT server.

NSLOOKUP Subcommand



Purpose

The NSLOOKUP statement specifies that resolution of the client connection address should be done. Once resolved, the results are saved in global variables and traced if TRACE ALL or TRACE NSLOOKUP is on.

Operands

YES

Indicates that a DNS lookup should be performed against the client IP address. The resulting host name and IP address are saved in the GLOBALV variables HOSTNAME and HOSTADDR. If TRACE ALL or TRACE NSLOOKUP is active, the results are added to the trace output as well. This is the default.

NO

Indicates that no DNS lookup of the client IP address should be performed.

EXIT *exit_name*

Indicates that a system administrator-provided exit exec should be called to determine what client IP address verification should be done. Based on the return code from the exec, the connection can be verified, not verified, or rejected. See [“DNS Lookup Exit” on page 655](#) for details on the interface to this exit. The exec name defaults to **UFTNSLKX**.

ON

Indicates to start exit invocation for all client resolution activity. This is the default.

OFF

Indicates to stop exit invocation for client resolution activity.

The NSLOOKUP command always resets any existing settings. For example, issuing NSLOOKUP YES after NSLOOKUP EXIT has the effect of turning off the exit and turning on UFT client verification.

QUERY Subcommand



Purpose

Use the QUERY subcommand to display configuration, user exit and tracing status.

Operands

All

Displays all available configuration, server exit and tracing status.

Exits

Displays status on all server exits, whether active or not, and the name of the exit routines in use (if applicable).

Trace

Displays status for all trace modes.

QUIT Subcommand

» QUIT «

Purpose

Use the QUIT subcommand (or its synonym STOP) to terminate the server.

Operands

The QUIT subcommand has no operands.

STOP Subcommand

» STOP «

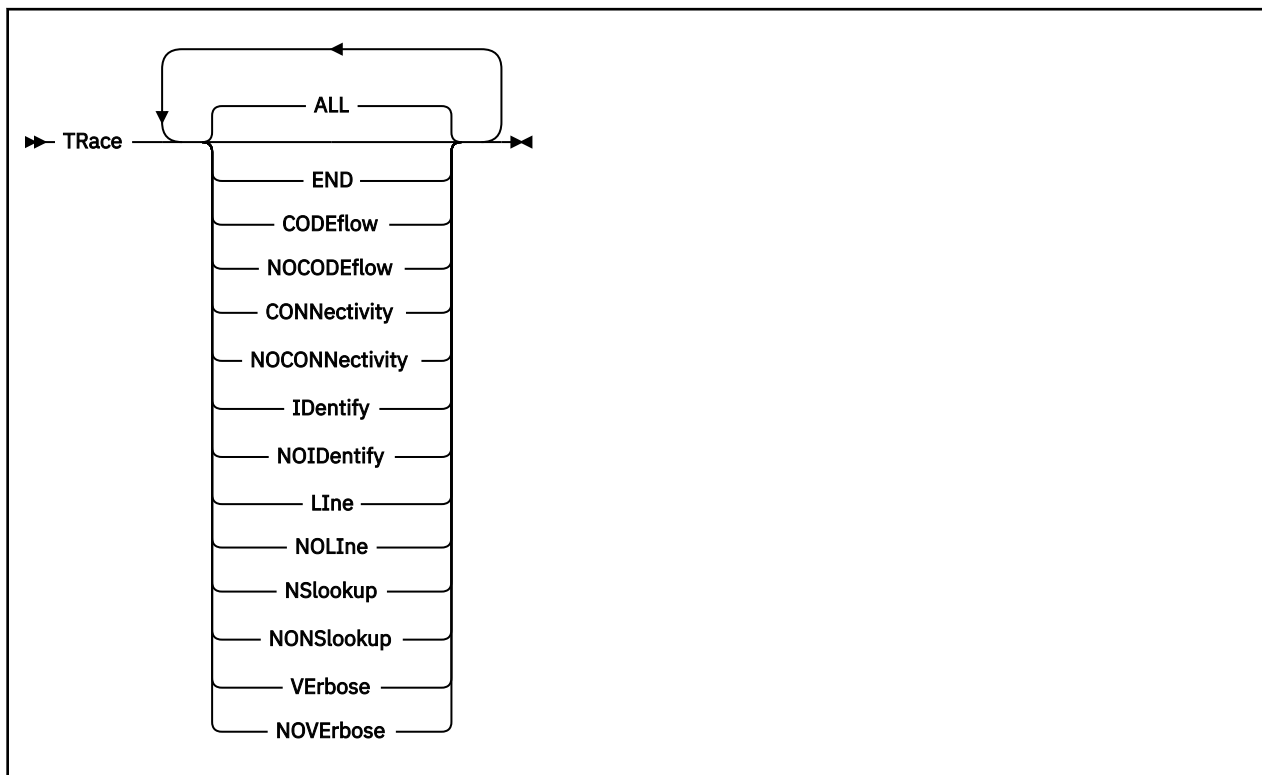
Purpose

Use the STOP subcommand (or its synonym QUIT) to terminate the server.

Operands

The STOP subcommand has no operands.

TRACE Subcommand



Purpose

Use the TRACE subcommand to activate or deactivate tracing within the UFT server. Trace records are placed in the UFT DEBUG file.

Note: If TRACE statements are found in the UFT configuration file, the active tracing is the sum of those traces and any traces activated using the TRACE subcommand.

Operands

ALL

Initiates tracing of all types.

END

Terminates tracing of all types.

CODEflow

Initiates tracing of UFT code flow.

NOCODEflow

Terminates tracing of UFT code flow.

CONNectivity

Initiates tracing of connectivity activity.

NOCONNectivity

Terminates tracing of connectivity activity.

IDENTify

Initiates tracing of the IDENTIFY function and results.

NOIDENTify

Terminates tracing of the IDENTIFY function and results.

LIne

Initiates tracing of all commands and replies and their associated connection number.

NOLine

Terminates tracing of all commands and replies and their associated connection number.

NSlookup

Initiates tracing of the resolver function and results.

NONSllookup

Terminates tracing of the resolver function and results.

VERBose

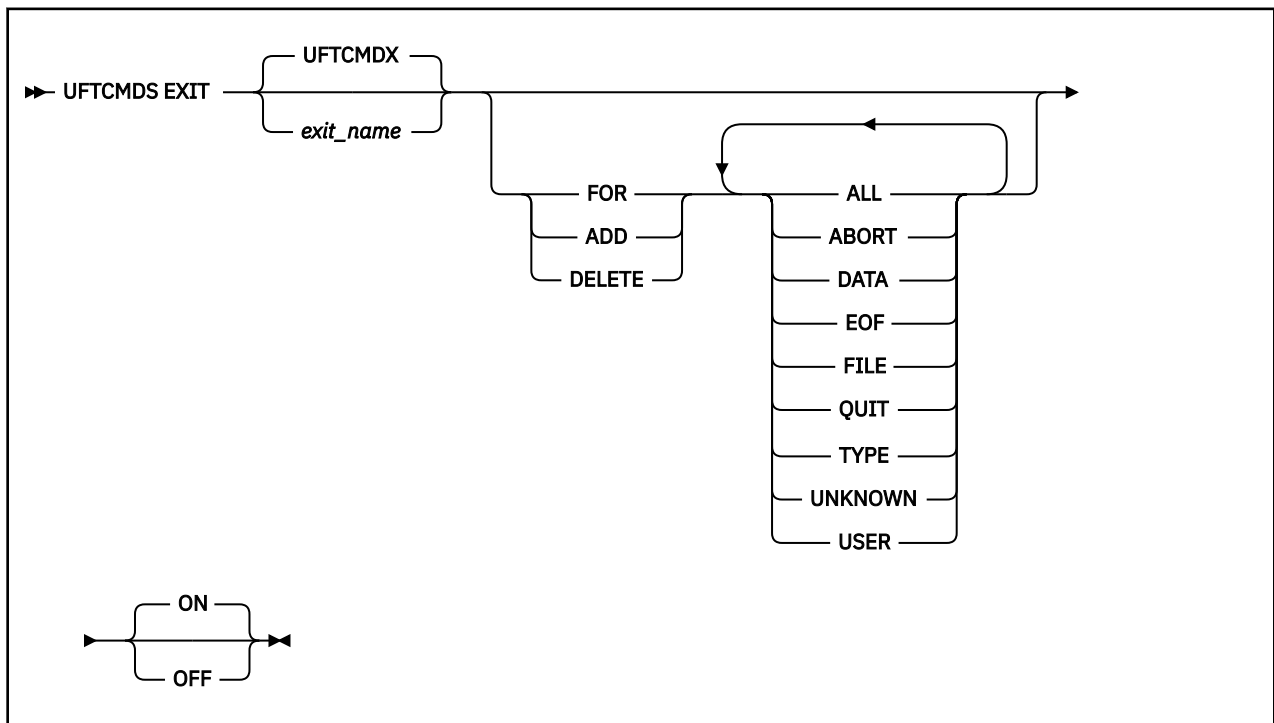
Initiates tracing of a collection of UFT data, equivalent to specifying CODEflow, CONNectivity and LINE.

NOVERBose

Terminates tracing of a collection of UFT data, equivalent to specifying NOCODEflow, NOCONNectivity and NOLINE.

Multiple TRACE subcommands or trace operands may be used to start more than one trace activity.

UFTCMDS EXIT Subcommand

**Purpose**

The UFTCMDS subcommand can be used to add or remove protocol command exit points on the running system, or it can be used to enable or disable command exit processing completely. For example, if the UFTCMDX exit exec is active for the DATA and FILE commands, the UFTCMDS EXIT UFTCMDX FOR TYPE ON subcommand will add TYPE to the already active command exit points.

Operands**exit_name**

Name of the exec to be called to handle any initiated protocol command exits. This exec is called on receipt of any commands for which the exit is enabled. See [“Protocol Commands Exit” on page 655](#) for details on the exit interface. The default exec name is **UFTCMDX**.

FOR

Defines which UFT protocol commands are enabled for the exit. If the UFTCMDS exit was previously defined, any commands specified with FOR replace all the previously enabled commands.

ADD

Defines which UFT protocol commands are to be added to any existing commands. The enabled commands will be the sum of any previously enabled commands and the commands specified with ADD.

DELETE

Defines which UFT protocol commands are to be deleted from exit processing. The exit remains active for any remaining enabled commands, if any.

command name

The *command name*, which consists of ALL, ABORT, DATA, EOF, FILE, QUIT, TYPE, UNKNOWN, and USER, identifies which protocol commands cause the exit to be called. For any individual command, any valid representation of the command or its synonyms, causes the exit to be called. Two keywords are not commands, but have special meaning. ALL indicates the exec is to be called for all UFT commands eligible for exit processing, and UNKNOWN indicates the exit is to be called for any unknown commands received.

ON

Indicates the exit exec should be invoked for any enabled UFT commands. This is the default.

OFF

Indicates the exit function should be initialized, but not yet invoked, for any UFT protocol commands. A UFTCMDS EXIT ON command is then required to initiate exit processing of UFT protocol commands.

UFT Clients and Servers for Other Platforms

To learn more about UFT clients and servers available for other systems, see [TCP/IP for z/VM \(https://www.ibm.com/vm/related/tcpip\)](https://www.ibm.com/vm/related/tcpip).

Chapter 18. Configuring the RSCS UFT Client

The RSCS Unsolicited File Transfer (UFT) Client provides support for the UFTASYNC option of the CMS SENDFILE command. This enables the file to be transmitted to the remote system when possible, without tying up the user's CMS session. The UFT-type link handles *one* file at a time, although successive files may be delivered to different hosts and user IDs. The sample RSCS configuration file has defined eight UFT-type links for handling multiple outgoing UFT data streams.

These steps describe how to configure an RSCS UFT-type link:

RSCS UFT Client Configuration Steps
1. Update the RSCSTCP CONFIG configuration file. 2. Update the RSCSUFT CONFIG configuration file. 3. Update the TCPIP DATA file.

Step 1: Update the RSCSTCP CONFIG Configuration File

The RSCSTCP CONFIG configuration file contains statements you can use to define your RSCS network. This file is read during initialization of the RSCS virtual machine. If this file is not found, RSCS initialization will fail. This file is the main configuration file for the RSCS server and will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. For more information, see [Chapter 12, “Configuring the RSCS Print Server,”](#) on page 353.

If eight UFT-type links are not enough to handle outgoing client requests, more can be defined by:

- Duplicating LINKDEFINE and PARM statement pairs
- Changing the link name to a unique one
- Adding the link name to the end of the GROUP statement

UFT Client LINKDEFINE and PARM Statements

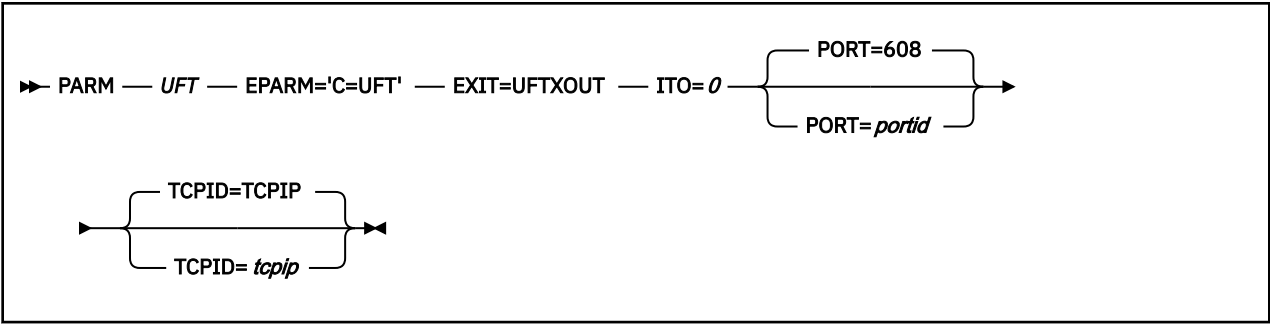
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an UFT link.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

```
➤➤ LINKDEFine — UFT — AST — FOrM * — TYPE UFT ➤➤
```

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The UFT keyword is in the *linkid* position, defining the name of the UFT link. This name must be unique on each LINKDEFINE statement. The *linkid* is a 1- to 8-character name of an UFT link that connects your local RSCS server to a remote UFT daemon in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.



Operands

- UFT**
Specifies the name provided on a LINKDEFINE statement.
- PORT=portid**
Specifies the port number on the remote host to connect to; defaults to 608 if not specified.
- TCPID=tcPIP**
Specifies the name of the TCPIP virtual machine; defaults to TCPIP if not specified.

Step 2: Update the RSCSUFT CONFIG Configuration File

The RSCS UFT configuration file contains statements you can use to further define how UFT client links handle file processing transmission to a daemon. This file will be stored on the TCP/IP customization minidisk, TCPMAINT 198. This allows you to:

- Change the default table used for EBCDIC to ASCII translation of the data
- Change the default table used for EBCDIC to ASCII translation of UFT commands
- Supply a name used as the owning user ID of the file instead of the user ID of the file originator

An asterisk (*) in column one denotes a comment line. Any line that does not have an asterisk in column one will be interpreted as a configuration entry. All entries must be capitalized.

- OWNERNAME=string**
Specifies an owning user ID name, up to 32 characters, for the OWNER UFT command. This statement can be used to provide a name other than the name of the file originator.
- TOASCII=string**
Provides a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCII= records to replace the 256-byte translation table.
- TOASCIIIC=string**
Provides a table for EBCDIC to ASCII translation of UFT commands, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple TOASCIIIC= records to replace the 256-byte translation table.

Step 3: Update the TCPIP DATA File

The TCPIP data file **must** contain these **uncommented** UFTserverID statements. These two entries enable you to send the files asynchronously using the RSCS server and receive them using the standard UFTD server.

Statement	Description
UFTserverID RSCS	Specifies the user ID the SENDFILE command will spool files to when using the UFTASYNC option. The default of * can be used as long as RSCS is specified in the SYSTEM NETID file.

Statement	Description
UFTserverID UFTD	This is used by the PEEK and RECEIVE commands when receiving UFT files to indicate the local UFT server user ID.

For more information on the PEEK, RECEIVE and SENDFILE commands, see *z/VM: CMS Commands and Utilities Reference*.

Chapter 19. 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 50 on page 670](#). 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.

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 675](#).

Table 48. 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 the you to change the translation table that is used. This is done using either client command line options, server initialization parameters, or configuration file statements. For example, the CMS FTP client provides a TRANSLATE option that you can use to provide the name of a translation table.

If no such option or configuration statement is provided, the clients and servers will use a *preferred* translation table that is specific to a particular client or server. If the preferred table cannot be found, the common *standard* translation will be used. The standard translation is loaded from STANDARD TCPxxBIN, if it is available, or from an equivalent that is compiled into the program.

Table 49 on page 668 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 49. Preferred Translation Tables

Program	Option	Preferred Translation Table
SMTP Server	None ¹	SMTP
SMTP Client (SENDFILE)	TRANSLATE table_name ^{2,3}	STANDARD
FTP Server	SITE TRANSLATE table_name ^{2,4}	SRVRFTP
FTP Client	TRANSLATE table_name ²	FTP
UFT Client (SENDFILE)	TRANSLATE table_name ²	STANDARD
UFT Server	TRANSLATE table_name ²	STANDARD
LPR Client	TRANSLATE table_name ²	LPR
NFS Server	XLATE=table_name ²	VMNFS

Table 49. Preferred Translation Tables (continued)

Program	Option	Preferred Translation Table
REXECD Server	None	REXECD
TELNET Client	TRANSLATE <i>table_name</i> ²	TELNET
TELNET Server (line mode)	None	STLINMOD

Note¹: For SMTP, an additional translation table may be specified for 8-bit MIME support. The *z/VM: TCP/IP Planning and Customization* contains more information in the "8BITMIME Statement" section.

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.

Note⁴: SITE TRANSLATE is a command, issued by the FTP client, that tells the FTP server which translation table to use for the current session. For more information, see *z/VM: TCP/IP User's Guide*.

If a table is explicitly referenced by a program option or configuration statement, the program will display an error message and stop if the translation table file cannot be found or loaded.

The file type of the translation table depends on whether any DBCS features are used. If Kanji translation is requested, the file type is TCPKJBIN. If Korean translation is requested, the file type is TCPHGBIN. For traditional Chinese, the file type is TCPCHBIN.

The *z/VM: TCP/IP User's Guide* contains information on the TRANSLATE option for the TCP/IP clients, and the *z/VM: TCP/IP Planning and Customization* contains information for the servers. See the *z/VM: CMS Commands and Utilities Reference* for information about the SENDFILE command.

Special Telnet Requirements

Telnet Client

The Telnet client requires a translation table that is different from the default table, STANDARD. The preferred translation table is provided by IBM as TELNET TCPXLBIN. If this file is not found, however, the default table will be used.

Country-specific translation tables for the Telnet application are provided. These tables have the file type TELXLATE. You must rename the selected file to TELNET TCPXLATE before it is converted to binary using the CONVXLAT command. The resulting TELNET TCPXLBIN should then be copied to TCPMAINT 592, replacing the IBM version.

Note: You cannot use the Telnet translation tables to change the LineFeed (X'0A') character.

Telnet Server

For line mode Telnet sessions, translation is performed by the Telnet server using the STANDARD translation table. If this table does not meet your needs, you can create an STLINMOD TCPXLATE table, convert it to binary using CONVXLAT, and copy the resulting STLINMOD TCPXLBIN file to the TCP/IP customization disk, TCPMAINT 198.

IBM-Supplied Translation Tables

In order to meet the translation needs of users and installations worldwide, more than 200 translation tables are provided with TCP/IP for your use. Some tables already exist in binary form; others require conversion using the CONVXLAT command, as described in ["Converting Translation Tables to Binary" on page 675](#).

Using Translation Tables

In order to make an IBM-supplied translation table the default translation table for a particular client or server program, store a copy of that translation table (in binary form) under the preferred translation table name for that program, as specified in [Table 49 on page 668](#).

Table 50. 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
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

Table 50. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
Belgium	BELGIAN	TCPXLATE	500	850
Canada	CANADIAN	TCPXLATE	37	850
Denmark and Norway	DANNOR	TCPXLATE	277	850
Netherlands	DUTCH	TCPXLATE	37	850
Finland and Sweden	FINSWED	TCPXLATE	278	850
France	FRENCH	TCPXLATE	297	850
Italy	ITALIAN	TCPXLATE	280	850
Japan	JAPANESE	TCPXLATE	281	850
OpenExtensions	POSIX	TCPXLATE	1047	819
Portugal	PORTUGUE	TCPXLATE	37	850
Spain and Spanish-speaking Latin America	SPANISH	TCPXLATE	284	850
Switzerland (French)	SWISFREN	TCPXLATE	500	850
Switzerland (German)	SWISGERM	TCPXLATE	500	850
United Kingdom	UK	TCPXLATE	285	850
United States	US	TCPXLATE	37	850
Standard (SBCS)	STANDARD	TCPXLATE	EBCDIC	ASCII
Standard Japanese Kanji	STANDARD	TCPKJLAT		
JIS X0208 1978			300	X0208
JIS X0208 1983			300	X0208
Shift JIS X0208			300	X0208
Extended Unix Code			300	EUC
IBM			300	300
Standard Korean Hangeul	STANDARD	TCPHGLAT		
KSC 5601 SBCS			833	1088
KSC 5601 DBCS			834	951
Hangeul SBCS			833	891
Hangeul DBCS			834	926
Standard Traditional Chinese	STANDARD	TCPCHLAT		
Traditional Chinese SBCS			037	904
Traditional Chinese DBCS			835	927

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

Note:

1. STANDARD TCPXLBIN is a 7-bit non-reversible translation table. For inbound data, all ASCII characters with values in the range X'80'-X'FF' will have the same translation as values X'00'-X'7F'. The high-order bit of each ASCII byte is ignored and is assumed to be zero. For example, ASCII X'B1' and X'31' will both be converted to EBCDIC X'F1'. For outbound data, EBCDIC control characters that do not have ASCII equivalents are converted to ASCII X'1A'. STANDARD should be used in situations where a client or server is known to treat the high-order bit in each byte as a parity bit.
2. All other SBCS translation tables provide a unique, one-to-one mapping of all 256 code points.
3. All tables translate ASCII LineFeed (LF, X'0A') to and from EBCDIC LF (X'25'), except for POSIX and 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 ;
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F ; B0 ;
7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 ; C0 ;
D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D ; D0 ;
79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96 ; E0 ;
97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07 ; F0 ;
;
; EBCDIC-to-ASCII table
;
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
00 01 02 03 1A 09 1A 7F 1A 1A 1A 0B 0C 0D 0E 0F ; 00 ;
10 11 12 13 1A 1A 08 1A 18 19 1A 1A 1C 1D 1E 1F ; 10 ;
1A 1A 1C 1A 1A 0A 17 1B 1A 1A 1A 1A 1A 05 06 07 ; 20 ;
1A 1A 16 1A 1A 1E 1A 04 1A 1A 1A 1A 14 15 1A 1A ; 30 ;
20 A6 E1 80 EB 90 9F E2 AB 8B 9B 2E 3C 28 2B 7C ; 40 ;
26 A9 AA 9C DB A5 99 E3 A8 9E 21 24 2A 29 3B 5E ; 50 ;
2D 2F DF DC 9A DD DE 98 9D AC BA 2C 25 5F 3E 3F ; 60 ;
D7 88 94 B0 B1 B2 FC D6 FB 60 3A 23 40 27 3D 22 ; 70 ;
F8 61 62 63 64 65 66 67 68 69 96 A4 F3 AF AE C5 ; 80 ;
8C 6A 6B 6C 6D 6E 6F 70 71 72 97 87 CE 93 F1 FE ; 90 ;
C8 7E 73 74 75 76 77 78 79 7A EF C0 DA 5B F2 F9 ; A0 ;
B5 B6 FD B7 B8 B9 E6 BB BC BD 8D D9 BF 5D D8 C4 ; B0 ;
7B 41 42 43 44 45 46 47 48 49 CB CA BE E8 EC ED ; C0 ;
7D 4A 4B 4C 4D 4E 4F 50 51 52 A1 AD F5 F4 A3 8F ; D0 ;
5C E7 53 54 55 56 57 58 59 5A A0 85 8E E9 E4 D1 ; E0 ;
30 31 32 33 34 35 36 37 38 39 B3 F7 F0 FA A7 FF ; F0 ;
```

Syntax Rules for SBCS Translation Tables

- Blanks are used only as delimiters for readability purposes.

- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).

Customizing DBCS Translation Tables

Each DBCS translation table file contains more than one translation table. TCPHGLAT and TCPHGBIN, for example, contain EBCDIC to ASCII and ASCII to EBCDIC translation tables for both the KSC 5601 and Hangeul PC code pages.

The standard DBCS binary tables are used by the FTP server, SMTP server, and FTP client programs.

The figures on the following pages show examples of the standard source for the Kanji, Hangeul, and Traditional Chinese DBCS translation tables.

These source files contain two column pairs for each code page. The first column pair specifies double-byte EBCDIC to ASCII code point mappings for the indicated code page. The second column pair specifies double-byte ASCII to EBCDIC code point mappings for the indicated code page.

Existing code point mappings may be changed by simply overwriting the existing hexadecimal code. New code point mappings may be specified by adding a new column pair with two double-byte hexadecimal codes. Code point mappings that are not specified, and are within the valid range for the code page, default to the "undefined" character, X'FFFF'.

The source file format allows EBCDIC to ASCII and ASCII to EBCDIC mappings to be specified separately. When adding or changing a code point mapping, care should be taken to modify both mappings for the code point. If, for example, a new mapping is added for EBCDIC to ASCII only, the ASCII to EBCDIC mapping for that code point will be the "undefined" character.

Any new code point mappings added outside the valid range for the corresponding code page will not be used by the programs that load the binary table.

DBCS Translation Table

The DBCS translation tables also contain SBCS code point mappings. These are used for mixed-mode DBCS strings, containing both SBCS and DBCS characters. Shift-out (X'0E') and shift-in (X'0F') characters are used on the EBCDIC host to denote the beginning and end of DBCS characters within a mixed-mode string.

The DBCS source files must contain exactly 256 SBCS code point mappings, situated at the end of the table. These may be modified to contain the required hexadecimal value.

Syntax Rules for DBCS Translation Tables

- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).
- Code point mappings in the file are position dependent. The first non-comment line for the DBCS and SBCS tables in the file will be used to establish the column position of the code point mappings, and must contain a conversion pair for each code page. Any conversion pairs on following lines must use the same column positions.
- It is permissible to leave blanks for code point mappings after the first line in the DBCS and SBCS areas. For example, if a line contains only one conversion pair, the column position will be used to determine which code page it refers to.
- The first column of each code page column pair, the "code index", must be in ascending numerical order. Any gaps in the ascending order will be marked as "undefined" in the binary table created by CONVXLAT.

Sample DBCS Translation Tables

The following examples are from the STANDARD DBCS translation table source files. Because these files are very large, only excerpts from the tables are shown. Ellipses (...) are used to indicate that information has been deleted.

Japanese Kanji DBCS Translation Tables

```
;
; STANDARD TCPKJLAT - Japanese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPKJBIN
; from this source file.
;
; DBCS Area - SJISETA,SJISATE
;             - JDECETA,JDECATE not used for STANDARD TCPKJBIN generation.
;
; SJISETA      SJISATE      JIS78ETA      JIS78ATE      JIS83ETA      JIS83ATE      ...
;
4040 8140 8140 4040 4040 2121 2121 4040 4040 2121 2121 4040 ...
4141 83BF 8141 4344 4141 2641 2122 4344 4141 2641 2122 4344 ...
4142 83C0 8142 4341 4142 2642 2123 4341 4142 2642 2123 4341 ...
4143 83C1 8143 426B 4143 2643 2124 426B 4143 2643 2124 426B ...
4144 83C2 8144 424B 4144 2644 2125 424B 4144 2644 2125 424B ...
4145 83C3 8145 4345 4145 2645 2126 4345 4145 2645 2126 4345 ...
4146 83C4 8146 427A 4146 2646 2127 427A 4146 2646 2127 427A ...
4147 83C5 8147 425E 4147 2647 2128 425E 4147 2647 2128 425E ...
4148 83C6 8148 426F 4148 2648 2129 426F 4148 2648 2129 426F ...
4149 83C7 8149 425A 4149 2649 212A 425A 4149 2649 212A 425A ...
;
;
; SBCS Area
;
; -----TCPKJBIN generation (no codefiles)-----|-----
; SJISETA ATE JIS78ETA ATE JIS83ETA ATE SJEUCETA ATE J7KETA J7KATE
;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
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 ..
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 ..
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 ..
09 1A 09 05 09 1A 09 05 09 1A 09 05 09 1A 09 05 ..
;
```

Hangeul DBCS Translation Tables

```
;
; STANDARD TCPHGLAT - Korean translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; use CONVXLAT to generate STANDARD TCPHGBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID - 951      Code Page ID - 926
; KSCETA      KSCATE      HANETA      HANATE
;
4040 A1A1 8FA1 D541 4040 8140 8140 4040
4141 A1A2 8FA2 D542 4141 8141 8141 4141
4142 A1A3 8FA3 D543 4142 8142 8142 4142
4143 A1A4 8FA4 D544 4143 8143 8143 4143
4144 A1A5 8FA5 D545 4144 8144 8144 4144
4145 A1A6 8FA6 D546 4145 8145 8145 4145
4146 A1A7 8FA7 D547 4146 8146 8146 4146
```

```

4147 A1A8 8FA8 D548      4147 8147 8147 4147
4148 A1A9 8FA9 D549      4148 8148 8148 4148
4149 A1AA 8FAA D54A      4149 8149 8149 4149
:
;
; SBCS Area
;
; Code Page ID 1088      Code Page ID 891
; SKSCETA   SKSCATE     SHANETA   SHANATE
;
00 00      00 00      00 00      00 00
01 01      01 01      01 01      01 01
02 02      02 02      02 02      02 02
03 03      03 03      03 03      03 03
04 FF      04 37      04 FF      04 37
05 09      05 2D      05 09      05 2D
06 FF      06 2E      06 FF      06 2E
07 1C      07 2F      07 1C      07 2F
08 FF      08 16      08 FF      08 16
09 FF      09 05      09 FF      09 05
:

```

Traditional Chinese DBCS Translation Tables

```

;
; STANDARD TCPCHLAT - Traditional Chinese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPCHBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID 927
; TCHETA      TCHATE
;
4040 8140      8140 4040
4141 83BF      8141 4344
4142 83C0      8142 4341
4143 83C1      8143 426B
4144 83C2      8144 424B
4145 83C3      8145 4345
4146 83C4      8146 427A
4147 83C5      8147 425E
4148 83C6      8148 426F
4149 83C7      8149 425A
:
;
; SBCS Area
;
; STCHETA     STCHATE
;
00 00      00 00
01 01      01 01
02 02      02 02
03 03      03 03
04 cf      04 37
05 09      05 2d
06 d3      06 2e
07 7f      07 2f
08 d4      08 16
09 d5      09 05
:

```

Converting Translation Tables to Binary

The CONVXLAT command converts a translation table source file to a binary file that is usable by TCP/IP client and server programs. CONVXLAT can be used to convert SBCS and DBCS source files.

For more information, see: [CONVXLAT Command](#).

Chapter 20. Testing and Verification

This section describes statements that can be used to verify the TCP/IP server is managing network traffic according to the established TCP/IP protocols.

Loopback Testing

In order to test your local machine, an address is reserved that always refers to your local host rather than any other hosts on a network. For IPv4, this class A network address is anything in the 127.x.x.x range. For IPv6, the reserved loopback address is ::1. You can also specify LOOPBACK as the host name. The loopback address can be used to test your local TCP/IP host without accessing the network. Loopback can be used with FTP, TELNET, SMTP, PORTMAP, and most other commands that accept an IP address. When you issue a command with the loopback address, the command is sent from your local host client to the local TCP/IP host where it is recognized as a loopback address and is sent to your local host server.

Using a loopback address on commands allows you to test client and server functions on the same host for proper operation.

TCP/IP Checksum Testing

In order to provide basic protection against transmission errors, TCP/IP uses checksums in its headers. The checksum in the IP header of an IPv4 packet covers only the bytes in the IPv4 header, whereas the checksum in the ICMP, IGMP, UDP, and TCP headers cover the header and the data (note that IPv6 does not compute a checksum on the header). The checksum is calculated by the sender using a specific algorithm. It is then stored in the header and sent as part of the datastream. The receiving side calculates the checksum on the data that is received using the same algorithm as the sender and compares its value to the checksum passed in the header. If the values do not match, the packet is rejected.

For debugging purposes, the TCP/IP server allows checksum verification to be turned on and off, but only at the TCP layer. The NOCHECKSUM statement can be specified (either in the TCP/IP configuration file, or through the use of the NETSTAT OBEY command) to temporarily disable TCP checksum testing on incoming packets. This might allow you to debug transmission errors on a noisy network, or to find a gateway or router that is losing bits in the transmission.

CHECKSUM Statement

The CHECKSUM statement is a TCP/IP configuration file statement that instructs the TCPIP virtual machine to reenale TCP checksum testing on incoming messages, if it has been disabled by the NOCHECKSUM statement.

►► CHECKSUM ◄◄

The TCP/IP module's default configuration is to verify checksums, so the CHECKSUM statement is rarely needed.

The CHECKSUM statement has no operands.

NOCHECKSUM Statement

The NOCHECKSUM statement is a TCP/IP configuration file statement that instructs the TCPIP virtual machine to ignore TCP checksum errors on incoming datagrams. Checksums are generated for outgoing datagrams, regardless of whether the NOCHECKSUM statement is used.

NOCHECKSUM should be used only for debugging purposes. It can affect data integrity, if used for normal operations.

➤ NOCHECKSUM ➤

To restore checksum validation, use the CHECKSUM statement.

The NOCHECKSUM statement has no operands.

Chapter 21. Using Source Code Libraries

TCP/IP source code is located (by default) on the 5VMTCP30 2B3 minidisk, whereas TCP/IP object code is located on the 5VMTCP30 2B2 minidisk.

Much of TCP/IP is coded in the Pascal language. Several components are written in C language or assembler. The following convention is used for file types:

Code	Description
ASSEMBLE	Assembly language program files
C	C language program files
COPY	Pascal language include files or assembler DSECTS
H	C language include files
MACRO	Assembler macro files
PASCAL	Pascal language program files
CSQL, SQC	C program files with SQL preprocessor statements.

To write application programs that interface with TCP/IP, use the following EXECs, as appropriate:

- VMFASM EXEC, VMFHASM EXEC, or VMFHLASM EXEC
- VMFPAS EXEC
- VMFC EXEC
- TCPTXT EXEC
- TCPLOAD EXEC
- TCPCOMP EXEC.

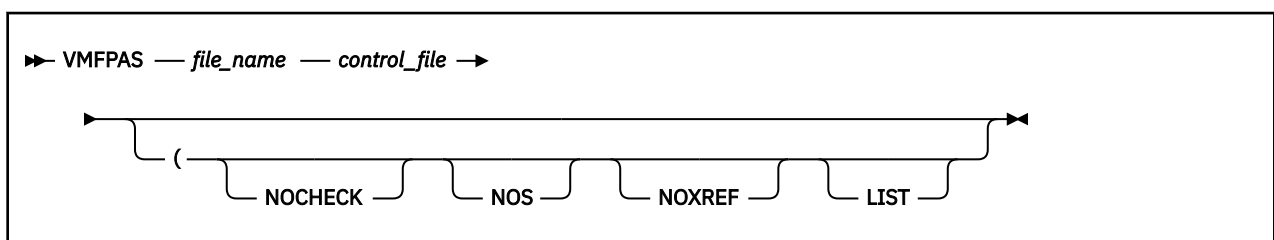
Note: The VMFASM, VMFHASM, and VMFHLASM execs are part of the VMSES/E component of z/VM. The other execs are part of TCP/IP.

These EXECs are explained in the following sections.

VMFASM EXEC, VMFHASM EXEC, and VMFHLASM EXEC

Use the VMFASM EXEC, VMFHASM EXEC, or VMFHLASM EXEC to update and compile assembler source code. For more information about these commands, see the [z/VM: VMSES/E Introduction and Reference](#).

VMFPAS EXEC



Purpose

The VMFPAS EXEC is similar in purpose to the VMSES/E VMFASM EXEC, but uses the CMS UPDATE command to apply updates to Pascal source and then compiles that source using the VS Pascal compiler.

Operands

file_name

Specifies the file name of the Pascal source program. The file type must be PASCAL.

control_file

Specifies the name of a control file that is used to apply the updates. The file type must be CNTRL.

NOCHECK

Causes compilation without Pascal runtime checking. You should use this command to achieve better performance in fully debugged code.

NOS

Does not produce a source listing.

NOXREF

Does not produce a cross-reference.

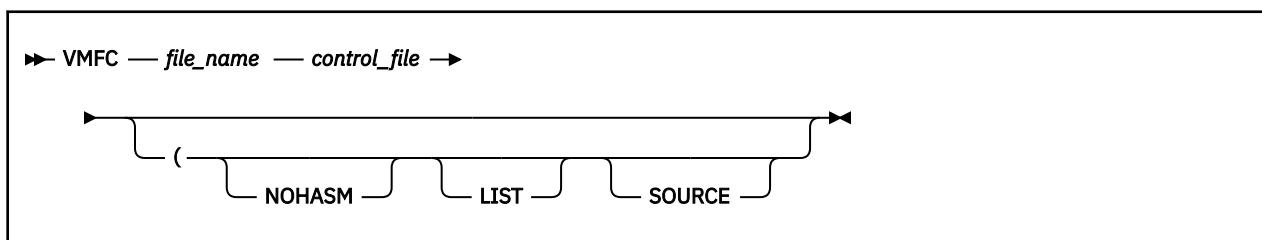
LIST

Produces an assembler listing.

Usage Notes

- VMFPAS does not exploit the VMSES/E service environment. For example, VMSES/E Software Inventory Files are not used during VMFPAS processing, and the object files produced by VMFPAS do not comply with VMSES/E naming conventions.
- The Pascal programs distributed with TCP/IP have been compiled with VS Pascal Compiler and Library 1.2 (5668-767). You must use this compiler to ensure successful compilation and compatibility with the distributed code.

VMFC EXEC



Purpose

The VMFC EXEC is similar in purpose to the VMSES/E VMFASM EXEC, but uses the CMS UPDATE command to apply updates to C source and then compiles that source using the CC exec provided with the IBM C for VM/ESA compiler.

Operands

file_name

Specifies the file name of the C source program. The file type must be C.

control_file

Specifies the file name of a control file that is used to apply the updates. The file type must be CNTRL.

NOHASM

Invokes assembler-XF. The default invokes the assembler specified at installation time.

LIST

Produces an assembler listing.

SOURCE

Produces a source listing.

Usage Notes

- VMFC does not exploit the VMSES/E service environment. For example, VMSES/E Software Inventory Files are not used during VMFPAS processing, and the object files produced by VMFPAS do not comply with VMSES/E naming conventions.
- The C programs distributed with TCP/IP have been compiled with IBM C for VM/ESA Compiler 3.1 (5654-033) and IBM Language Environment for MVS & VM 1.8 (5688-198).

TCPTXT EXEC

►► TCPTXT — *load_list* — *control_file* ◄◄

Purpose

The TCPTXT EXEC constructs a text library (TXTLIB) when it is given a list of text file names and a control file.

Almost all TXTLIBs used by TCP/IP are built using VMSES/E. If you have to build one that is not, you can use the TCPTXT command. Before running the command, make sure the latest serviced level of each text deck included in the text library is available, with the appropriate file type.

Use TCPTXT to construct the XMLIB TXTLIB.

Operands***load_list***

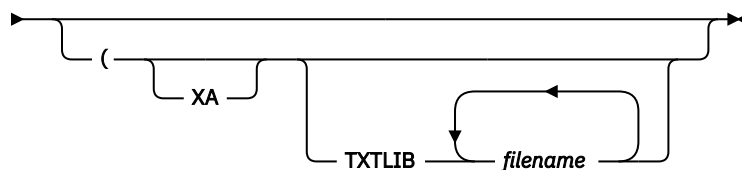
Specifies a file name with file type LOADLIST that contains the text file names to be included in the TXTLIB.

control_file

Specifies the file name of a control file that is used to select the text files. The file type must be CNTRL.

TCPLOAD EXEC

►► TCPLOAD — *load_list* — *control_file* — *type* →

**Purpose**

When running TCPLOAD, you must access all disks containing object files as extensions of the A disk. The TCPLOAD EXEC generates a module when given a list of text file names and a control file.

Almost all MODULEs used by TCP/IP are built using VMSES/E. If you have to build one that is not, you can use the TCPLOAD command. Before running the command, make sure the latest serviced level of each text deck included in the text library is available, with the appropriate file type.

Operands

load_list
Specifies a file name with a file type of LOADLIST that contains text file names to be included in the load module. The first line in the *load_list* specifies the name of the main object module. Subsequent lines specify additional object modules to be included in the load module.

control_file
Specifies the file name of the control file that is used to select the text files. The file type must be CNTRL.

type
Specifies the programming language.

C
Includes SCEELKED, CMSLIB, TCPASCAL, and TCPLANG TXTLIBs.

C-ONLY
Includes SCEELKED and CMSLIB TXTLIBs.

PASCAL
Includes TCPASCAL and TCPLANG TXTLIBs.

Note: The COMMTXT TXTLIB is always included in the GLOBAL TXTLIB statement.

XA
Includes the options AMODE 31 RMODE ANY on the LOAD and GENMOD commands. It also includes the TCPXXA TXTLIB.

TXTLIB filename
Allows you to specify up to 50 TXTLIBs that are added to the GLOBAL TXTLIB command. RPC users should specify TXTLIB RPCLIB.

Usage Notes

- The generated module has the same file name as the file name of the load list. The load lists shipped with TCP/IP are:

MODULE	DISK
SAMPLE_C LOADLIST	TCPMAINT 592
SAMPLE_S LOADLIST	TCPMAINT 592

- You can ignore the following message from the preloading step:

DMSPRE236E UNRESOLVED EXTERNAL REFERENCE(S) ENCOUNTERED

TCPCOMP EXEC

```
➡ TCPCOMP — load_list — control_file →
```

Purpose

The TCPCOMP EXEC recompiles or assembles all the files listed in a LOADLIST file. This command is useful when you need to recompile all the files of a component.

Operands

load_list

Specifies a file that has a file type of LOADLIST or LKEDCTRL that contains the file names to be recompiled or assembled.

control_file

Specifies the file name of a control file that is used to apply the updates. The file type must be CNTRL.

Special Considerations

The following are special considerations that apply to rebuilding the TCP/IP code:

- To reassemble CMMALLOC and CMQSTOR, use either the H Assembler or HL Assembler.
- To reassemble NSDBSQL ASMSQL, link to the SQL system minidisk and execute the SQLPP exec.

Appendix A. Using TCP/IP with an External Security Manager

Some of the TCP/IP servers are involved in making system resources available to clients. These servers ensure that the clients' credentials (user ID and password) are valid, and they ensure that the client accesses only those resources permitted by those credentials.

Credential validation and resource access are under the control of CP or an external security manager (ESM). IBM's Resource Access Control Facility (RACF) is an example of an external security manager that offers effective user authentication, resource access controls, and logging capabilities.

The following servers can be configured to interface with an ESM to provide system resource protection, if desired:

- FTP (FTPSERVE, CSMSEVER)
- NFS (VMNFS)
- REXEC (REXECD)

This appendix describes the interfaces used by TCP/IP servers to access z/VM security-related services, and the specific actions that must be taken if you use RACF.

If you use an external security manager other than RACF, consult the appropriate publications for your security manager for similar configuration information.

Server Validation Methods

Password phrase support is not available for all servers. Because of this, certain servers use a traditional validation method by invoking DMSPWCHK or RPIVAL, in accordance with the `:ESM_Enable.` tag in DTCPARMS.

Servers that support password phrases use an enhanced validation method by invoking DMSPASS. If the ESM does not support use of DIAGNOSE X'88' subcode 8 (via DMSPASS), the RPIVAL command is used as a fallback. When RPIVAL is used, `:ESM_Enable.` is required only when `:ESM_Validate.` is not RPIVAL or when authentication rates are such that additional performance is desired (from pre-loading the `:ESM_Validate.` module into memory).

Table 51 on page 685 indicates each server's validation method as either tradition or enhanced.

Table 51. Server validation methods	
Server	Method
FTP	Enhanced
LP	Traditional
NFS	Traditional
REXEC	Enhanced

Security Interfaces

All password validation, "Logon By" permission verification, and minidisk access control services are obtained through these CSL routines:

DMSPASS

Provides password, password phrase, and "Logon By" verification.

DMSESM

Provides information that is used with DMSLINK and DMSPWCHK.

DMSPWCHK

Provides password and "Logon By" verification.

DMSLINK

Provides minidisk and virtual reader access services.

These CSL routines may access ESM services using the following interfaces:

RPIVAL

A command that determines if a VM user ID and password are valid.

RACROUTE

A macro that is used to verify "Logon By" privileges and to determine the permitted level of access to minidisks and virtual readers.

The *z/VM: CMS Callable Services Reference* contains detailed information on the operation of these CSL routines, including their interactions with the ESM.

These interfaces provide two levels at which you can implement your own security scheme:

- You can replace one or more of the CSL routines with those of your own creation, or those provided by another vendor, or
- You can provide your own password validation program to perform the function of RPIVAL (if used), and a RACROUTE request handler.

In the descriptions that follow, it is assumed that the CSL routines provided by IBM are being used and that the default tag values in the IBM DTCPARMS file have not been overridden or changed.

Server Initialization

When :ESM_Enable.YES is specified in the DTCPARMS file for any of the servers listed under [Appendix A, "Using TCP/IP with an External Security Manager,"](#) on page 685, two things will happen when the server is started:

1. The RACROUTE macro request handler is enabled using the command identified on the :ESM_Racroute. tag in the DTCPARMS file (RPIUCMS or RPIDUMY, by default).

The operation of RPIUCMS is defined in the *z/VM: Security Server RACROUTE Macro Reference*. RPIDUMY is included with TCP/IP and is used in cases where a server requires password validation services, but does not use RACROUTE.

2. The FTP and NFS servers will call DMSESM to obtain a security token that may be used to uniquely identify the server to the ESM.

The security token is included on calls to DMSPWCHK and/or DMSLINK. These routines use the security token to determine whether their services should be provided using native CP functions or those of the ESM.

Client Authentication

Whenever a client requests access to the system, the server will call DMSPASS or DMSPWCHK to ensure that the user ID and password provided are valid. Expired passwords may not be used to access the system and the servers do not provide a mechanism for clients to change the password.

If the "Logon By" form of FTP, NFS, or REXEC client login is used, both user ID and password are passed to DMSPASS or DMSPWCHK, which will ensure that the client's own user ID and password are valid, and that the user has permission to "logon by" to the specified user ID (called the *target ID*).

When an ESM is in use, DMSPWCHK will verify the client's user ID and password using the command identified on the :ESM_Validate. tag in the DTCPARMS file. The default command is RPIVAL.

Resource Access

Whenever an FTP or NFS client requests access to a minidisk or virtual reader, the server calls DMSLINK. This routine determines if the client may access the requested resource.

For minidisks, DMSLINK also obtains a link to a minidisk. Minidisk passwords may or may not be required, depending on your external security manager configuration. If a password is required, but one was not provided, DMSLINK will indicate this condition and the servers will act accordingly. In the case of FTP, the client is prompted to issue the ACCT subcommand to provide a minidisk password. For NFS, the mount request is rejected, requiring the client to provide the password using the MDISKPW= parameter on the mount string.

Note: Incorrect passwords provided by FTP or NFS clients are not tracked or journaled by CP.

For a virtual reader, DMSLINK is called to determine whether the client may view and manipulate the contents of the reader queue (DMSLINK RC=0), or only send files to the queue (DMSLINK RC=4).

The DTCPARMS File

Three DTCPARMS file entries relate to TCP/IP's use of external security manager interfaces:

:ESM_Enable.

Indicates whether an External Security Manager (ESM) is to be used to authenticate and authorize access to resources managed by this server. When no value is specified for this tag the default is NO, in which case native CP security services (diagnose X'88') will be used.

:ESM_Validate.

Identifies a program to validate user IDs and passwords supplied by clients. When no value is specified for this tag the default is RPIVAL. This routine is used only when the ESM does not support DIAGNOSE X'88' subcode 8 or the server uses DMSPWCHK to verify passwords.

If a different program is used, it must follow the programming conventions (parameter format and return codes) used by RPIVAL. More information on the RPIVAL command may be found in [z/VM: RACF Security Server Macros and Interfaces](#).

:ESM_Racroute.

Identifies a program to initialize and terminate the RACROUTE environment. When no value is specified for this tag the default is RPIUCMS, a program provided by the RACF product.

:ESM_Racroute.RPIDUMY should be specified whenever you choose to use ESM password validation, but want to provide native CP minidisk protection. RPIDUMY provides a non-zero security token on the DMSESM call, but will defer to CP whenever a RACROUTE authorization call is made.

The default values for the :ESM_Validate. and :ESM_Racroute. tags are appropriate for RACF. If an ESM other than RACF is in use, it may be necessary to define alternate values for the :ESM_Validate. and :ESM_Racroute. tags.

These tags may be specified on the :Type.Server entry for a server, or they may be specified on the appropriate :Type.Class entry in the DTCPARMS file. Providing this information at the class level ensures that all servers of the same class will use the specified ESM services.

If you choose to modify the class entries, copy them from IBM DTCPARMS to a local DTCPARMS file — do not modify IBM DTCPARMS because it may be replaced by the application of service or the upgrade to a new release of z/VM. With each service application or upgrade, you will need to verify that any changes introduced by IBM are incorporated into your local DTCPARMS file.

The suggested alternative is to use the global server profile exit (TCPRUNXT EXEC) to set these tags. By using the profile exit, you may make decisions based on server user ID or server class. See [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33 for more information on using profile exits.

For more information about using these tags, see [“DTCPARMS Tags”](#) on page 37.

Minidisk Security

The FTP and NFS servers link to minidisks on behalf of clients. They do not have the same function as a traditional "batch" machine and, therefore, should not have the same minidisk security policy as a batch machine.

To access minidisks, the FTP and NFS servers use `diagnose X'D4'` to change their identity. It is important that your ESM implement a security policy for these servers that makes minidisk access decisions based solely on this *alternate user ID*, not the user ID of the server virtual machine (FTPSEVE, for example).

Failure to implement this security policy could result in unintended or undesirable access by a client to the FTP or NFS server's 191 disks or to TCP/IP configuration files which may contain sensitive data.

Using TCP/IP with RACF

The following procedure enables the server machines listed under [Appendix A, "Using TCP/IP with an External Security Manager,"](#) on page 685, to use the authorization interfaces provided by RACF. If you use a security manager product other than RACF, consult the appropriate product publications.

Steps for using TCP/IP with RACF

Before you begin: It will simplify TCP/IP server administration if you create one or more RACF groups that contain some or all of the TCP/IP servers that are installed. By doing so, you can grant a new server all necessary permissions by simply connecting it to the appropriate group(s).

Perform the following steps to use TCP/IP with RACF:

1. If the RACF ICHRCX02 installation exit is installed, either modify it to not allow an alternate user to access resources that can be accessed by the FTP and NFS servers or, if your installation does not use CMS batch-style virtual machines, remove ICHRCX02 from your RACF configuration. For more information, see [z/VM: RACF Security Server System Programmer's Guide](#) and the previous section, "Minidisk Security" on page 688.
2. Identify the RACF/VM service machine to which RACROUTE requests will be sent. Ensure the dispatcher virtual machine has access to the RACF SERVMACH file. For more information, see [z/VM: RACF Security Server System Programmer's Guide](#).
3. Enable RACF/VM profile protection for `DIAGNOSE X'88'`:
 - a. Confirm that there are no members called `DIAG088/NOCTL` in the active `VMXEVENT` profile.
 - b. Create a profile called `DIAG088` in the `VMCMD` class with a default access of `NONE`:

```
RDEFINE VMCMD DIAG088 UACC(NONE)
```

- c. Ensure that the `VMCMD` class is active:

```
SETROPTS CLASSACT(VMCMD)
```

Note: If you do not enable RACF profile protection for `DIAGNOSE X'88'`, the FTP, NFS, and REXEC servers must be defined with `OPTION DIAG88` in their directory entries.

4. Give each FTP, NFS, and REXEC server permission to perform password validation using `DMSPASS` (which uses `DIAGNOSE X'88'` subcode 8):

```
PERMIT DIAG088 CLASS(VMCMD) ID(FTPSEVE VMNFS REXECD) ACCESS(READ)
```

For more information, see [z/VM: RACF Security Server Security Administrator's Guide](#).

5. Authorize the FTP and NFS servers to use RACROUTE services.
 - a. Confirm the RACFVM user ID service machine contains `IUCV ALLOW` in its CP directory entry to provide `IUCV` authorization for the FTP and NFS servers.

Note: Adding IUCV ALLOW to the CP directory entry grants authorization to any user who can connect to the system; however, authorization to perform RACROUTE functions requires explicit RACF authorization as granted in the following steps. For additional information about IUCV statements, see *z/VM: CP Planning and Administration*.

- b. Create a profile named ICHCONN in the FACILITY class.

```
RDEFINE FACILITY ICHCONN UACC(NONE)
```

- c. Give UPDATE access authority to the appropriate service machines.

```
PERMIT ICHCONN CLASS(FACILITY) ID(FTPSERVE VMNFS REXECD) ACCESS(UPDATE)
```

Note: Update access to profile ICHCONN allows the dispatcher service machine to issue limited RACROUTE requests.

- d. Activate the FACILITY class if it is not already active.

```
SETOPTS CLASSACT(FACILITY)
```

6. Permit the FTP and NFS servers to access resources on behalf of clients:

- Use the FTPPERM and/or NFSPERM commands for users with discrete VMBATCH profiles.
- Give the FTP and NFS servers CONTROL access to a generic VMBATCH profile so users without a discrete VMBATCH profile may use FTP and NFS services:

```
PERMIT ** CLASS(VMBATCH) ID(FTPSERVE VMNFS REXECD) ACCESS(CONTROL)
```

Note: If some users have discrete profiles and others do not, you can use both of these techniques. For additional information, see “FTPPERM and NFSPERM Commands” on page 689.

7. Modify the DTCPARMS server entries for all servers listed under [Appendix A, “Using TCP/IP with an External Security Manager,”](#) on page 685 to enable the needed RACF interfaces.

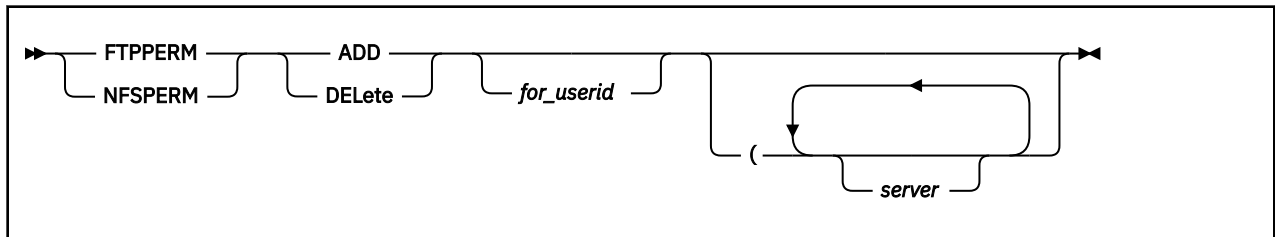
```
:ESM_Enable.YES
```

Note: You may find it more convenient to make these changes on a class basis, instead of an individual server basis, to ensure that all servers of the same class will have the same security characteristics.

The global profile exit, TCPRUNXT EXEC, provides an ideal way to set these tags. See [Chapter 5, “General TCP/IP Server Configuration,”](#) on page 33 for more information on using profile exits.

8. Give the TCP/IP service user ID UPDATE permission to all server A-disks as well as TCPMAINT 591, 592, and 198. To find the TCP/IP service user ID, refer to *Program Directory for TCP/IP for z/VM*.

FTPPERM and NFSPERM Commands



Purpose

The FTPPERM and NFSPERM commands are used to allow the FTP and NFS servers to access all of the disks that a user would have access to if that user were logged on. This is accomplished by granting the servers CONTROL access to the user's VMBATCH profile.

If a user does not have a discrete VMBATCH profiles, FTPPERM and NFSPERM will issue a warning message. In this case, the FTP and NFS servers must instead be given CONTROL access to a generic VMBATCH profile.

Where discrete profiles exist, FTPPERM and NFSPERM must be run by each user who chooses to use FTP or NFS. Alternatively, the system administrator may issue these commands on the users' behalf.

Operands

Parameter	Description

ADD	Permits the FTP or NFS servers to act on the user's behalf when accessing minidisks.
DELeTe	Removes permission for the FTP or NFS servers to act on the user's behalf when accessing minidisks.
for_userid	Is the VM user ID on whose behalf this command is being issued. This parameter is used only by the system administrator.
server	The list of FTP or NFS server virtual machines that are affected. If no list is provided, the default is obtained from the FTP_SERVERS or NFS_SERVERS GLOBALV variable in the TCPIP group. If no GLOBALV variable has been set, the default is FTPSERVE for FTPPERM, and VMNFS for NFSPERM. It may be useful to set these GLOBALV variables in the SYSPROF EXEC so that users do not need to know the names of all of the servers that are present in your environment.

Examples

The following example gives FTPSERVE and FTPSERV2 permission to act on user ALAN's behalf when accessing minidisks.

```
ftpperm add alan (ftpserv ftpserv2
Ready;
```

Because NFSPERM and FTPPERM perform the same function, you can use one command to give permission to FTP and NFS servers at the same time. This following example gives both FTPSERVE and VMNFS the needed permissions.

```
globalv select tcpip set1 ftp_servers FTPSERVE VMNFS
Ready;

ftpperm add
Ready;
```

Appendix B. SMF records

SMF Record Type 83, subtype 3 records

When auditing is enabled for LDAP events, SMF record type 83, subtype 3 records are recorded in the SMF file. Each logged LDAP event has a unique event code with a corresponding event code qualifier. The event qualifier indicates whether the event succeeded or failed. For more information on SMF records and the relocates that are common to all SMF Type 83 subtype 2 and above records, see [z/VM: RACF Security Server Macros and Interfaces](#).

Table 52 on page 691 shows LDAP event codes and event code qualifiers:

Table 52. LDAP event codes				
Event	Operation	Event qualifier	Description	Relocate type sections
1	Add	0	Successful LDAP operation	Common relocates, (100-114) 201-208
		1	Failed LDAP operation	
2	Bind	0	Successful LDAP operation	Common relocates, (100-114) 201-207 209
		1	Failed LDAP operation	
3	Compare	0	Successful LDAP operation	Common relocates, (100-114) 201-207 210
		1	Failed LDAP operation	
4	Connect	0	Successful LDAP operation	Common relocates, (100-114) 201-207
		1	Failed LDAP operation	
5	Delete	0	Successful LDAP operation	Common relocates, (100-114) 201-207
		1	Failed LDAP operation	
6	Disconnect	0	Successful LDAP operation	Common relocates, (100-114) 201-207
		1	Failed LDAP operation	
7	Extended Operation	0	Successful LDAP operation	Common relocates, (100-114) 201-207 221
		1	Failed LDAP operation	
8	Modify	0	Successful LDAP operation	Common relocates, (100-114) 201-207 211-213
		1	Failed LDAP operation	
9	Modify DN	0	Successful LDAP operation	Common relocates, (100-114) 201-207 214-216
		1	Failed LDAP operation	
10	Search	0	Successful LDAP operation	Common relocates, (100-114) 201-207 218-220
		1	Failed LDAP operation	
11	Unbind	0	Successful LDAP operation	Common relocates (100-114) 201-207
		1	Failed LDAP operation	
12	Abandon	0	Successful LDAP operation	Common relocates (100-114) 201-207, 222
		1	Failed LDAP operation	

Table 53 on page 692 shows LDAP extended relocates:

Table 53. LDAP extended relocates					
Relocate #	Field name	Type	Length	Audited by event code	Description
100	LDAP_RESERVED_01	CHAR	16	N/A	Reserved.
101	LDAP_REQ_TIMESTP	CHAR	16	1, 2, 3, 5, 7, 8, 9, 10, 11	The elapsed time for request completion (in microseconds).
102	LDAP_SERVER_URL	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
103	LDAP_CONN_ID	CHAR	8	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Internal Connection ID. Used to indicate operations performed on the same connection.
104	LDAP_MESSAGE_ID	CHAR	8	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Message ID from BER. Used to connect events
105	LDAP_BIND_DN	CHAR	512	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Bind DN for the connection
106	LDAP_CLIENT_SECL	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	LDAP client security label
107	LDAP_SRC_IP_ADDR	CHAR	16	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
108	LDAP_AUDIT_VERSION	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Version of the LDAP audit support in use
109	LDAP_EVENT_CODE	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Event number 1 - 11
110	LDAP_RESERVED_04	CHAR	227	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Reserved
111	LDAP_MAPCERT_OPT	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	sslMapCertificate configuration option (not supported on z/VM)
112	LDAP_MAPPED_SAFID	CHAR	8	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	SAF ID (if any) associated with the bind DN
113	LDAP_POLICY_UPDATED	CHAR	2	2	T (true) or F (false) if the password policy operational attribute values in the bind DN entry are updated.
114	LDAP_FLAGS	CHAR	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Character representation of the hex value for flag settings. See note "1" on page 693 for more information.
201	LDAP_PROTOCOL_VER	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Request version 2 or 3
202	LDAP_RETURN_CODE	INT	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Return code in hex (blank if no return code was provided)
203	LDAP_ERROR_MSG	CHAR	256	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Reason code message number and text
204	LDAP_ENTRY_NM	CHAR	512	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Target entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
205	LDAP_RESERVED_05	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Reserved
206	LDAP_ENTRY_SUFFIX	CHAR	64	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Suffix of the DN
207	LDAP_CNTRLS_PRESENT	CHAR	512	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Control OID - Criticality pairs for each control

Table 53. LDAP extended relocates (continued)					
Relocate #	Field name	Type	Length	Audited by event code	Description
208	LDAP_ADD_ATTR	CHAR	64 (maximum 1299)	1	Name of attribute added Will have one of these for each unique attribute added (with a maximum of 20, note allowing 1 space in between each value)
209	LDAP_BIND_MECH	CHAR	16	2	Simple, external, DIGEST-MD5, CRAM-MD5
210	LDAP_COMPARE_ATTR	CHAR	64	3	Attribute being compared
211	LDAP_MOD_ATTR_DEL	CHAR	64 (maximum 909)	8	Name of an attribute that was deleted. Will have one of these for each unique attribute (with a maximum of 14, note allowing 1 space in between each value)
212	LDAP_MOD_ATTR_ADD	CHAR	64 (maximum 909)	8	Name of an attribute that was added. Will have one of these for each unique attribute (with a maximum of 14, allowing 1 space in between each value)
213	LDAP_MOD_ATTR_REP	CHAR	64 (maximum 909)	8	Name of an attribute to that was replaced. Will have one of these for each unique attribute (with a maximum of 14, note allowing 1 space in between each value)
214	LDAP_MDN_NEW_RDN	CHAR	32	9	New relative distinguished name for target entry
215	LDAP_MDN_DOLD_RDN	CHAR	1	9	T (true) or F (false), should the old RDN be deleted
216	LDAP_MDN_NEW_SUP	CHAR	512	9	New parent of target entry
218	LDAP_SEARCH_FILTER	CHAR	256	10	Search filter attribute
219	LDAP_SEARCH_SCOPE	CHAR	12	10	Subtree search, base search,
220	LDAP_SEARCH_ATTRS	CHAR	64	10	Attribute requested to be returned on the search. One of these for each
221	LDAP_XOP_REQ_NAME	CHAR	64	11	OID for extended operation
222	LDAP_TARGET_MESSAGE_ID	CHAR	8	12	The abandon target message ID from the BER encoded data from the client that is used to match the abandon and the request that is being abandoned.
223	LDAP_DISCONNECT_CAUSE	CHAR	10	6	Indicates the reason of a disconnect event. See note “2” on page 694 for more information.

Notes:

1. The LDAP_FLAGS value is the character representation of the hex value for the following possible flag settings:

ADMIN_ROLE_NONE	0x00000001
ADMIN_ROLE_ROOT	0x00000002
ADMIN_ROLE_DIR	0x00000004
ADMIN_ROLE_REPL	0x00000008
ADMIN_ROLE_SCHEMA	0x00000010
ADMIN_ROLE_CONFIG	0x00000020
ADMIN_ROLE_PASSWORD	0x00000040
ADMIN_ROLE_OPER	0x00000080
ADMIN_ROLE_SAF	0x00000100

where, the bound user has one or more of the following administrative roles,

```
ADMIN_ROLE_NONE=No administrator
ADMIN_ROLE_ROOT=Root administrator
ADMIN_ROLE_DIR=Directory data administrator
ADMIN_ROLE_REPL=Replication administrator
ADMIN_ROLE_SCHEMA=Schema administrator
ADMIN_ROLE_CONFIG=Server configuration group member
ADMIN_ROLE_PASSWORD=Password administrator
ADMIN_ROLE_OPER=Operational administrator
ADMIN_ROLE_SAF=The administrative user is a member of
cn=safadmingroup,cn=configuration and the roles are defined in RACF.
```

Example: If the user is a member of **cn=safadmingroup,cn=configuration** and has been permitted to all administrative roles, 0x000001FF is in this field.

2. The LDAP_DISCONNECT_CAUSE value is the character representation of the hex value for the following possible flag settings:

```
DISCONNECT_TIMEOUT                0x00000001
DISCONNECT_NIC_FAILED             0x00000002
DISCONNECT_CLOSE_ABNORMAL        0x00000004
DISCONNECT_CLOSE_CLIENT          0x00000008
DISCONNECT_MESSAGE_INVALID       0x00000010
DISCONNECT_ERROR_INTERNAL        0x00000020
DISCONNECT_ERROR_WRITE           0x00000040
DISCONNECT_ERROR_SSL             0x00000080
DISCONNECT_BLOCKED_TIMEOUT       0x00000100
```

where,

```
DISCONNECT_TIMEOUT=Idle connection timeout.
DISCONNECT_NIC_FAILED=Network interface failed.
DISCONNECT_CLOSE_ABNORMAL=Connection closed abnormally.
DISCONNECT_CLOSE_CLIENT=Client closed connection.
DISCONNECT_MESSAGE_INVALID=The request is not a valid LDAP
message.
DISCONNECT_ERROR_INTERNAL=Internal error. Such as, out of storage
error, unable to encode or decode LDAP message.
DISCONNECT_ERROR_WRITE=Failed to write LDAP message to network
connection.
DISCONNECT_ERROR_SSL=Error occurred in System SSL.
DISCONNECT_BLOCKED_TIMEOUT=Blocked connection time out.
```

Note: There might be two values that are combined. For example: DISCONNECT_TIMEOUT and DISCONNECT_CLOSE_CLIENT can occur simultaneously and the value is 0x00000009.

RACF SMF unload utility output

A general description of the format of the tabular records created by the SMF unload utility can be found in *z/VM: RACF Security Server Macros and Interfaces*, in the topic "The format of the unloaded SMF type83 data".

The data following the header in the tabular record varies according to the LDAP event that was recorded. The LDAP event types are:

Table 54. Event type strings	
Event code from SMF type 83 subtype 3 records	Tabular output Event type strings
1	*ADD
2	*BIND
3	*COMPARE
4	*CONN
5	*DELETE

Table 54. Event type strings (continued)	
Event code from SMF type 83 subtype 3 records	Tabular output Event type strings
6	*DISCONN
7	*EXOP
8	*MODIFY
9	*MODDN
10	*SEARCH
11	*UNBIND
12	*ABANDON

Event qualifiers are the same for all LDAP event codes:

Table 55. Event qualifiers		
Event qualifier	Event qualifier number	Event description
SUCCESS	0	Audit of successful LDAP operation
FAILURE	1	Audit of failed LDAP operation

Table 56. Event specific fields for LDAP add event (Event code 1)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 1 (add)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00

Table 56. Event specific fields for LDAP add event (Event code 1) (continued)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRL_PRESNT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)
LDAP_ADD_ATTR	CHAR	64 (maximum 1299)	5324	6622	Name of attribute added. Will have one of these for each unique attr added. (with a maximum of 20, note allowing 1 space in between each value)

Table 57. Event specific fields for LDAP bind event (Event code 2)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection

Table 57. Event specific fields for LDAP bind event (Event code 2) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 2 (bind)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Represents value of sslMapCertificate configuration option (not supported on z/VM)
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	T (true) or F (false) if the password policy operational attribute values in the bind DN entry are updated.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return Code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason Code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)
LDAP_BIND_MECH	CHAR	16	5324	5339	Simple, external, DIGEST-MD5, CRAM-MD5

Table 58. Event specific fields for LDAP compare event (Event code 3)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 3 (compare)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return Code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason Code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target Entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)

Table 58. Event specific fields for LDAP compare event (Event code 3) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_COMPARE_ATTR	CHAR	64	5324	5387	Attribute being compared.

Table 59. Event specific fields for LDAP connect, delete, disconnect, and unbind events (Event code 4, 5, 6, 11)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 4, 5, 6, 11 (connect, delete, disconnect, unbind)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return Code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason Code, message number and text

Table 59. Event specific fields for LDAP connect, delete, disconnect, and unbind events (Event code 4, 5, 6, 11) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target Entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)

Table 60. Event specific fields for LDAP extended operations event (Event code 7)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTAMP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 7 (extended operations)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved

Table 60. Event specific fields for LDAP extended operations event (Event code 7) (continued)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRL_ PRESNT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)
LDAP_XOP_REQ_NAME	CHAR	64	5324	5387	OID for extended operations

Table 61. Event specific fields for LDAP modify event (Event code 8)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 8 (modify)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN

Table 61. Event specific fields for LDAP modify event (Event code 8) (continued)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)
LDAP_MOD_ATTR_DEL	CHAR	64 (maximum 909)	5324	6233	Name of attribute that was deleted. Will have one of these for each unique attr (with a maximum of 14, note allowing 1 space in between each value).
LDAP_MOD_ATTR_ADD	CHAR	64 (maximum 909)	6236	7145	Name of an attribute that was added. Will have one of these for each unique attr (with a maximum of 14, note allowing 1 space in between each value).
LDAP_MOD_ATTR_REPLACE	CHAR	64 (maximum 909)	7148	8057	Name of an attribute that was replaced. Will have one of these for each unique attr (with a maximum of 145, note allowing 1 space in between each value).

Table 62. Event specific fields for LDAP modify DN event (Event code 9)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)

Table 62. Event specific fields for LDAP modify DN event (Event code 9) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 9 (modify DN)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return Code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason Code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target Entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)
LDAP_MDN_NEW_RDN	CHAR	32	5324	5355	New relative distinguished name for target entry.
LDAP_MDN_DOLD_RDN	CHAR	1	5358	5358	T (true) or F (false), should the old RDN be deleted.
LDAP_MDN_NEW_SUP	CHAR	512	5361	5872	New parent of target entry.

Table 63. Event specific fields for LDAP search event (Event code 10)					
Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	Time the request was received (in microseconds)
LDAP_SERVER_URL	CHAR	32	3036	3067	The auditing server's URL for the connection that the client came in on (Listen URL)
LDAP_CONN_ID	CHAR	8	3070	3077	Internal connection ID. Used to indicate operations performed on the same connection
LDAP_MESSAGE_ID	CHAR	8	3080	3087	Message ID from BER. Used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	Bind DN for the connection
LDAP_CLIENT_SECL	CHAR	32	3604	3635	LDAP client security label
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	IP of the client request (PC if request came across PC interface; SLAPI for SLAPI internal requests)
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	Version of the LDAP audit support in use
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 10 (search)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	SAF ID (if any) associated with the bind DN
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note “1” on page 693 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	Request version 2 or 3
LDAP_RETURN_CODE	INT	10	3926	3935	Return Code in hex (blank if no return code was provided)
LDAP_ERROR_MSG	CHAR	256	3938	4193	Reason Code, message number and text
LDAP_ENTRY_NM	CHAR	512	4196	4707	Target Entry of the operation (Bind DN, DN to be deleted, DN to be added, etc.)
LDAP_RESERVED_05	CHAR	32	4710	4741	Reserved
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	Suffix of the DN.
LDAP_CNTRLS_PRESENT	CHAR	512	4810	5321	Control OID - Criticality pairs for each control (Note, no value)

Table 63. Event specific fields for LDAP search event (Event code 10) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_SEARCH_FILTER	CHAR	256	5324	5579	Search filter attribute
LDAP_SEARCH_SCOPE	CHAR	12	5582	5593	Subtree search, base search
LDAP_SEARCH_ATTRS	CHAR	64	5596	5659	Attr requested to be returned on the search. One of these for each.

Appendix C. Activity Log Records

This section describes the fields that are present in the activity log when it is configured to log LDAP client requests.

Note: The activity log is not an official interface to the LDAP server. It may change at any time.

Activity Log Start and End Field Descriptions

Table 64 on page 707 describes the fields that are present in start or end activity log records. Activity log start records are logged when the **GLDLOG_TIME** environment variable is set to **notime**, or **logFileRecordType** is set to **begin** in the configuration file. Activity log start and end records are logged when the **GLDLOG_TIME** environment variable is set to **time** or **logFileRecordType** is set to **both** in the configuration file.

Table 64. Start or end activity log fields		
Field name	Operations	Description
attrs	search, add(V1+), modify(V1+)	The name of the attributes requested to be returned on the search request.
base	search	The base DN for the search request.

Table 64. Start or end activity log fields (continued)

Field name	Operations	Description																		
bindFlags	bind	The hex value for the following possible flag settings is: <table><tr><td>ADMIN_ROLE_NONE</td><td>0x00000001</td></tr><tr><td>ADMIN_ROLE_ROOT</td><td>0x00000002</td></tr><tr><td>ADMIN_ROLE_DIR</td><td>0x00000004</td></tr><tr><td>ADMIN_ROLE_REPL</td><td>0x00000008</td></tr><tr><td>ADMIN_ROLE_SCHEMA</td><td>0x00000010</td></tr><tr><td>ADMIN_ROLE_CONFIG</td><td>0x00000020</td></tr><tr><td>ADMIN_ROLE_PASSWORD</td><td>0x00000040</td></tr><tr><td>ADMIN_ROLE_OPER</td><td>0x00000080</td></tr><tr><td>ADMIN_ROLE_SAF</td><td>0x00000100</td></tr></table> where, ADMIN_ROLE_NONE No administrator ADMIN_ROLE_ROOT Root administrator ADMIN_ROLE_DIR Directory data administrator ADMIN_ROLE_REPL Replication administrator ADMIN_ROLE_SCHEMA Schema administrator ADMIN_ROLE_CONFIG Server configuration group member ADMIN_ROLE_PASSWORD Password administrator ADMIN_ROLE_OPER Operational administrator ADMIN_ROLE_SAF The administrative user is a member of cn=safadmingroup,cn=configuration and the roles are defined in RACF. Example: If the user is a member of cn=safadmingroup,cn=configuration and has been permitted to all administrative roles, bindFlags = 1FF is in this field.	ADMIN_ROLE_NONE	0x00000001	ADMIN_ROLE_ROOT	0x00000002	ADMIN_ROLE_DIR	0x00000004	ADMIN_ROLE_REPL	0x00000008	ADMIN_ROLE_SCHEMA	0x00000010	ADMIN_ROLE_CONFIG	0x00000020	ADMIN_ROLE_PASSWORD	0x00000040	ADMIN_ROLE_OPER	0x00000080	ADMIN_ROLE_SAF	0x00000100
ADMIN_ROLE_NONE	0x00000001																			
ADMIN_ROLE_ROOT	0x00000002																			
ADMIN_ROLE_DIR	0x00000004																			
ADMIN_ROLE_REPL	0x00000008																			
ADMIN_ROLE_SCHEMA	0x00000010																			
ADMIN_ROLE_CONFIG	0x00000020																			
ADMIN_ROLE_PASSWORD	0x00000040																			
ADMIN_ROLE_OPER	0x00000080																			
ADMIN_ROLE_SAF	0x00000100																			

Table 64. Start or end activity log fields (continued)

Field name	Operations	Description
cause	disconnect(V1+)	The reason code value for disconnect, the possible values, and description are: 1: Idle connection time-out. 2: Network interface failed. 4: Connection closed abnormally. 8: Client closed connection. 10: The request is not a valid LDAP message. 20: Internal error. For example: out of storage error, unable to encode or decode LDAP message. 40: Failed to write LDAP message to network connection. 80: Error occurred in System SSL. 100: Blocked connection time-out. For example, if the value is 5, it means that the disconnect cause is: (1) Idle connection time-out and (2) Connection closed abnormally.
connid	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, connect(V1+), disconnect(V1+), abandon(V1+), unknown(V1+)	The internal connection ID used to indicate operations performed.
count	search	The number of entries returned on the search request.
DN	add, bind, delete, disconnect(V1+), modify, modrdn, unbind	The target entry of the operation (base DN for the search, bind DN, DN to be deleted, DN to be added, and so on).
entry	compare	The target entry of the compare operation.
filter	search	The search filter specified in the search request.
IP	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, connect(V1+), disconnect(V1+), abandon(V1+), unknown(V1+)	The IP address of the client performing the request.
listen	bind, unbind, connect(V1+)	The listen URL that the LDAP server is listening on that received the client connection.
msgid	add(V1+), bind(V1+), compare(V1+), delete(V1+), exop(V1+), modify(V1+), search(V1+), abandon(V1+), unknown(V1+)	Message Identifier.

Field name	Operations	Description								
newRdn	modrdn	The new relative distinguished name for the target entry in the modrdn request.								
policyUpdated	bind	• T (true) or F (false) if the password policy operational attribute values in the bind DN entry are updated. • NA if the operation is compare and the attribute that is being compared is not userPassword.								
pReqOID	extendedop	The OID of the extended operation request.								
rc	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind	The LDAP server return code for the operation.								
safid	bind	The SAF ID, if any, associated with the bind DN.								
scope	search	The scope of the search request. It is set to 0 for subtree searches, 1 for one level searches, and 2 for subtree searches.								
searchFlags	search	The hex value for the following possible flag settings is: <table><tr><td>SEARCH_PAGE_FIRST</td><td>0x00000001</td></tr><tr><td>SEARCH_PAGE_NEXT</td><td>0x00000002</td></tr><tr><td>SEARCH_PAGE_LAST</td><td>0x00000004</td></tr><tr><td>SEARCH_SORT_COMPLETE</td><td>0x00000008</td></tr></table> where, SEARCH_PAGE_FIRST First page of a paged search request SEARCH_PAGE_NEXT Ensuing page of a paged search request SEARCH_PAGE_LAST Last page of a paged search request SEARCH_SORT_COMPLETE Sorted search requested and completed successfully Example: If the last page of a successfully paged and sorted search request has been returned to the client application, searchFlags = 0E is set in this field.	SEARCH_PAGE_FIRST	0x00000001	SEARCH_PAGE_NEXT	0x00000002	SEARCH_PAGE_LAST	0x00000004	SEARCH_SORT_COMPLETE	0x00000008
SEARCH_PAGE_FIRST	0x00000001									
SEARCH_PAGE_NEXT	0x00000002									
SEARCH_PAGE_LAST	0x00000004									
SEARCH_SORT_COMPLETE	0x00000008									
targetmsgid	abandon(V1+)	The target message ID of the abandon operation.								
type	unknown(V1+)	The type of the request.								

Note: Fields for the **logFileVersion 1** or above are indicated with (V1+).

Activity Log mergedRecord Field Descriptions

Table 65 on page 711 describes the fields that are present in mergedRecord activity log records. Activity log mergedRecords are logged when the **GLDLOG_TIME** environment variable is set to **mergedrecord** or **logFileRecordType** is set to **mergedRecord** in the configuration file.

Table 65. mergedRecord activity log fields		
Field name	Operations	Description
attr	compare	The name of the attribute being compared.
attrs	add(V1+), modify(V1+), search	The name of the attributes that are requested to be returned on the request.
bind	add, bind, compare, disconnect(V1+), delete, extendedop, modify, modrdn, search, unbind, unknown(V1+),	The bind DN for the connection.
bindFlags	bind	The hex value for the following possible flag settings is: ADMIN_ROLE_NONE 0x00000001 ADMIN_ROLE_ROOT 0x00000002 ADMIN_ROLE_DIR 0x00000004 ADMIN_ROLE_REPL 0x00000008 ADMIN_ROLE_SCHEMA 0x00000010 ADMIN_ROLE_CONFIG 0x00000020 ADMIN_ROLE_PASSWORD 0x00000040 ADMIN_ROLE_OPER 0x00000080 ADMIN_ROLE_SAF 0x00000100 where, ADMIN_ROLE_NONE No administrator ADMIN_ROLE_ROOT Root administrator ADMIN_ROLE_DIR Directory data administrator ADMIN_ROLE_REPL Replication administrator ADMIN_ROLE_SCHEMA Schema administrator ADMIN_ROLE_CONFIG Server configuration group member ADMIN_ROLE_PASSWORD Password administrator ADMIN_ROLE_OPER Operational administrator ADMIN_ROLE_SAF The administrative user is a member of cn=safadmingroup,cn=configuration and the roles are defined in RACF. Example: If the user is a member of cn=safadmingroup,cn=configuration and has been permitted to all administrative roles, bindFlags = 1FF is in this field.

Table 65. mergedRecord activity log fields (continued)

Field name	Operations	Description
cause	disconnect(V1+)	The reason code value for disconnect, the possible values, and description are: 1: Idle connection time-out. 2: Network interface failed. 4: Connection closed abnormally. 8: Client closed connection. 10: The request is not a valid LDAP message. 20: Internal error. For example: out of storage error, unable to encode or decode LDAP message. 40: Failed to write LDAP message to network connection. 80: Error occurred in System SSL. 100: Blocked connection time-out. For example, if the value is 5, it means that the disconnect cause is: (1) Idle connection time-out and (2) Connection closed abnormally.
clientIP	abandon(V1+), add, bind, compare, connect(V1+), delete, disconnect(V1+), extendedop, modify, modrdn, search, unbind, unknown(V1+)	The IP address of the client performing the request.
connid	abandon(V1+), add, bind, compare, connect(V1+), delete, disconnect(V1+), extendedop, modify, modrdn, search, unbind, unknown(V1+)	The internal connection ID used to indicate operations performed on the same client connection.
controls	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The OID of the control on the request. If there are multiple controls on the request, they are separated by ':'.
count	search	The number of entries returned on the search request.
delete oldRdn	modrdn	T (true) or F (false) if the old relative distinguished name (RDN) is deleted in the modrdn request.
filter	search	The search filter specified in the search request.
listen	bind, connect(V1+), unbind	The listen URL that the LDAP server was listening on that received the client connection.
mech	bind	Indicates the authentication method. It has one of the following values: SIMPLE , EXTERNAL , DIGEST-MD5 , or CRAM-MD5

Table 65. mergedRecord activity log fields (continued)		
Field name	Operations	Description
msgid	abandon(V1+), add(V1+), bind(V1+), compare(V1+), delete(V1+), exop(V1+), modify(V1+), search(V1+), unknown(V1+)	Message Identifier.
newRdn	modrdn	The new relative distinguished name for the target entry in the modrdn request.
policyUpdated	bind	• T (true) or F (false) if the password policy operational attribute values in the bind DN entry are updated. • NA if the operation is compare and the attribute that is being compared is not userPassword.
rc	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+),	The LDAP server return code for the operation.
rsn	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+),	The LDAP reason code message number and text, if there is one, sent from the server to the client.
saf	bind, unbind	The SAF ID, if any, associated with the bind DN.
scope	search	The scope of the search request. It is set to 0 for subtree searches, 1 for one level searches, and 2 for subtree searches.
searchFlags	search	The hex value for the following possible flag settings is: <pre> SEARCH_PAGE_FIRST 0x00000001 SEARCH_PAGE_NEXT 0x00000002 SEARCH_PAGE_LAST 0x00000004 SEARCH_SORT_COMPLETE 0x00000008 </pre> where, SEARCH_PAGE_FIRST First page of a paged search request SEARCH_PAGE_NEXT Ensuing page of a paged search request SEARCH_PAGE_LAST Last page of a paged search request SEARCH_SORT_COMPLETE Sorted search requested and completed successfully Example: If the last page of a successfully paged and sorted search request has been returned to the client application, searchFlags = 0E is set in this field.

<i>Table 65. mergedRecord activity log fields (continued)</i>		
Field name	Operations	Description
seclabel	bind, unbind	The LDAP client security label, if there is one.
target	add, compare, modify, modrdn, search	The target entry of the operation (base DN for the search, bind DN, DN to be deleted, DN to be added, and so on).
targetmsgid	abandon(V1+)	The target message ID of the abandon operation.
time	abandon(V1+), , add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+),	The elapsed time for request completion (in microseconds).
type	unknown(V1+)	The type of the request.
XOP	extendedop	The OID of the extended operation request.

Note: Fields for the **logFileVersion 1** or above are indicated with (V1+).

Appendix D. Related Protocol Specifications

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
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
1034	<i>Domain Names—Concepts and Facilities</i>	P.V. Mockapetris

RFC	Title	Author
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
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
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
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

RFC	Title	Author
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
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
1387	<i>RIP Version 2 Protocol Analysis</i>	G. Malkin
1389	<i>RIP Version 2 MIB Extension</i>	G. Malkin
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
1493	<i>Definition of Managed Objects for Bridges</i>	E. Decker, P. Langille, A. Rijssinghani, K. McCloghrie
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
1738	<i>Uniform Resource Locators (URL)</i>	T. Berners-Lee, L. Masinter, M. McCahill
1813	<i>NFS Version 3 Protocol Specification</i>	B. Callaghan, B. Pawlowski, P. Stauback, Sun Microsystems Incorporated
1823	<i>The LDAP Application Program Interface</i>	T. Howes, M. Smith
2460	<i>Internet Protocol, Version 6 (IPv6) Specification</i>	S. Deering, R. Hinden
2052	<i>A DNS RR for specifying the location of services (DNS SRV)</i>	A. Gulbrandsen, P. Vixie

RFC	Title	Author
2104	<i>HMAC: Keyed-Hashing for Message Authentication</i>	H. Krawczyk, M. Bellare, R. Canetti
2222	<i>Simple Authentication and Security Layer (SASL)</i>	J. Myers
2247	<i>Using Domains in LDAP/X.500 Distinguished Names</i>	S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri
2251	<i>Lightweight Directory Access Protocol (v3)</i>	M. Wahl, T. Howes, S. Kille
2252	<i>Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions</i>	M. Wahl, A. Coulbeck, T. Howes, S. Kille
2253	<i>Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names</i>	M. Wahl, S. Kille, T. Howes
2254	<i>The String Representation of LDAP Search Filters</i>	T. Howes
2255	<i>The LDAP URL Format</i>	T. Howes, M. Smith
2256	<i>A Summary of the X.500 (96) User Schema for use with LDAPv3</i>	M. Wahl
2279	<i>UTF-8, a transformation format of ISO 10646</i>	F. Yergeau
2373	<i>IP Version 6 Addressing Architecture</i>	R. Hinden, S. Deering
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
2713	<i>Schema for Representing Java Objects in an LDAP Directory</i>	V. Ryan, S. Seligman, R. Lee
2714	<i>Schema for Representing CORBA Object References in an LDAP Directory</i>	V. Ryan, R. Lee, S. Seligman
2732	<i>Format for Literal IPv6 Addresses in URLs</i>	R. Hinden, B. Carpenter, L. Masinter
2743	<i>Generic Security Service Application Program Interface Version 2, Update 1</i>	J. Linn
2744	<i>Generic Security Service API Version 2 : C-bindings</i>	J. Wray
2820	<i>Access Control Requirements for LDAP</i>	E. Stokes, D. Byrne, B. Blakley, P. Behera
2829	<i>Authentication Methods for LDAP</i>	M. Wahl, H. Alvestrand, J. Hodges, R. Morgan
2830	<i>Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security</i>	J. Hodges, R. Morgan, M. Wahl
2831	<i>Using Digest Authentication as a SASL Mechanism</i>	P. Leach, C. Newman
2849	<i>The LDAP Data Interchange Format (LDIF)</i>	G. Good

RFC	Title	Author
2873	<i>TCP Processing of the IPv4 Precedence Field</i>	X. Xiao, A. Hannan, V. Paxson, E. Crabble
3377	<i>Lightweight Directory Access Protocol (v3): Technical Specification</i>	J. Hodges, R. Morgan
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
4191	<i>Default Router Preferences and More-Specific Routes</i>	R. Draves, D. Thaler
4517	<i>LDAP Syntaxes and Matching Rules</i>	S. Legg
4523	<i>LDAP Schema Definitions for X.509 Certificates</i>	K. Zeilenga
5095	<i>Deprecation of Type 0 Routing Headers in IPv6</i>	J. Abley, P. Savola, G. Neville-Nei
5175	<i>IPv6 Router Advertisement Flags Option</i>	B. Haberman, R. Hinden
5722	<i>Handling of Overlapping IPv6 Fragments</i>	S. Krishnan
6946	<i>Processing of IPv6 "Atomic" Fragments</i>	F. Gont
6980	<i>Security Implications of IPv6 Fragmentation with IPv6</i>	F. Gont

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or on a subscription basis. Online copies are available using FTP from the NIC at `nic.ddn.mil`. Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

Where:

nnnn

Is the RFC number.

TXT

Is the text format.

PS

Is the PostScript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of RFC *nnnn* for text versions or a subject line of RFC *nnnn*.PS for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.

For more information, contact `nic@nic.ddn.mil`. Information is also available at [Internet Engineering Task Force](http://www.ietf.org) (www.ietf.org).

Appendix E. Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this book.

AIX®	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Program Interface
APPC	Advanced Program-to-Program Communications
APPN	Advanced Peer-to-Peer Networking
ARP	Address Resolution Protocol
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
AUI	Attachment Unit Interface
BFS	Byte File System
BIOS	Basic Input/Output System
BNC	Bayonet Neill-Concelman
CCITT	Comite Consultatif International Telegraphique et Telephonique. The International Telegraph and Telephone Consultative Committee
CLIST	Command List
CMS	Conversational Monitor System
CP	Control Program
CPI	Common Programming Interface
CREN	Corporation for Research and Education Networking
CSD	Corrective Service Diskette
CTC	Channel-to-Channel
CU	Control Unit
CUA	Common User Access
DASD	Direct Access Storage Device
DBCS	Double Byte Character Set
DLL	Dynamic Link Library
DNS	Domain Name System
DOS	Disk Operating System
DPI	Distributed Program Interface
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EISA	Enhanced Industry Standard Adapter
ESCON	Enterprise Systems Connection Architecture
FAT	File Allocation Table
FTAM	File Transfer Access Management

Abbreviations and Acronyms

FTP	File Transfer Protocol
FTP API	File Transfer Protocol Applications Programming Interface
GCS	Group Control System
GDF	Graphics Data File
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPL	Initial Program Load
ISA	Industry Standard Adapter
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IUCV	Inter-User Communication Vehicle
JES	Job Entry Subsystem
JIS	Japanese Institute of Standards
JCL	Job Control Language
LAN	Local Area Network
LAPS	LAN Adapter Protocol Support
LCS	IBM LAN Channel Station
LDAP	Lightweight Directory Access Protocol
LPQ	Line Printer Query
LPR	Line Printer Client
LPRM	Line Printer Remove
LPRMON	Line Printer Monitor
LU	Logical Unit
MAC	Media Access Control
Mbps	Megabits per second
MBps	Megabytes per second
MCA	Micro Channel Adapter
MIB	Management Information Base
MIH	Missing Interrupt Handler
MILNET	Military Network
MHS	Message Handling System
MTU	Maximum Transmission Unit
MVS	Multiple Virtual Storage
MX	Mail Exchange

NCP	Network Control Program
NDIS	Network Driver Interface Specification
NFS	Network File System
NIC	Network Information Center
NLS	National Language Support
NSFNET	National Science Foundation Network
OS/2	Operating System/2®
OSA	Open Systems Adapter
OSF	Open Software Foundation, Inc.
OSI	Open Systems Interconnection
OSIMF/6000	Open Systems Interconnection Messaging and Filing/6000
OV/MVS	OfficeVision/MVS
OV/VM	OfficeVision/VM
PAD	Packet Assembly/Disassembly
PC	Personal Computer
PCA	Parallel Channel Adapter
PDN	Public Data Network
PDU	Protocol Data Units
PING	Packet Internet Groper
PIOAM	Parallel I/O Access Method
POP	Post Office Protocol
PROFS	Professional Office Systems
PSCA	Personal System Channel Attach
PSDN	Packet Switching Data Network
PU	Physical Unit
PVM	Passthrough Virtual Machine
RACF	Resource Access Control Facility
RARP	Reverse Address Resolution Protocol
REXEC	Remote Execution
REXX	Restructured Extended Executor Language
RFC	Request For Comments
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RSCS	Remote Spooling Communications Subsystem
SAA	Systems Application Architecture®
SBCS	Single Byte Character Set
SFS	Shared File System

Abbreviations and Acronyms

SLIP	Serial Line Internet Protocol
SMIL	Structure for Management Information
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOA	Start of Authority
SPOOL	Simultaneous Peripheral Operations Online
SQL	IBM Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TSO	Time Sharing Option
TTL	Time-to-Live
UDP	User Datagram Protocol
VGA	Video Graphic Array
VM	Virtual Machine
VMCF	Virtual Machine Communication Facility
VM/ESA	Virtual Machine/Enterprise System Architecture
VMSES/E	Virtual Machine Serviceability Enhancements Staged/Extended
VTAM®	Virtual Telecommunications Access Method
WAN	Wide Area Network
XDR	eXternal Data Representation

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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

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

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

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

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

PI

<....Programming Interface information....>

PI end

Trademarks

IBM, the IBM logo, and [ibm.com](https://www.ibm.com/legal/copytrade)[®] are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on [IBM Copyright and trademark information](https://www.ibm.com/legal/copytrade) (<https://www.ibm.com/legal/copytrade>).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java[™] and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

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

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

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek

your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at [IBM Privacy Statement](https://www.ibm.com/privacy) (<https://www.ibm.com/privacy>)
- [Cookies and Similar Technologies](https://www.ibm.com/privacy#Cookies_and_Similar_Technologies) (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see [z/VM: General Information](#).

Where to Get z/VM Information

The current z/VM product documentation is available in [IBM Documentation - z/VM \(https://www.ibm.com/docs/en/zvm\)](https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- [z/VM: License Information](#), GI13-4377
- [z/VM: General Information](#), GC24-6286

Installation, Migration, and Service

- [z/VM: Installation Guide](#), GC24-6292
- [z/VM: Migration Guide](#), GC24-6294
- [z/VM: Service Guide](#), GC24-6325
- [z/VM: VMSES/E Introduction and Reference](#), GC24-6336

Planning and Administration

- [z/VM: CMS File Pool Planning, Administration, and Operation](#), SC24-6261
- [z/VM: CMS Planning and Administration](#), SC24-6264
- [z/VM: Connectivity](#), SC24-6267
- [z/VM: CP Planning and Administration](#), SC24-6271
- [z/VM: Getting Started with Linux on IBM Z](#), SC24-6287
- [z/VM: Group Control System](#), SC24-6289
- [z/VM: I/O Configuration](#), SC24-6291
- [z/VM: Running Guest Operating Systems](#), SC24-6321
- [z/VM: Saved Segments Planning and Administration](#), SC24-6322
- [z/VM: Secure Configuration Guide](#), SC24-6323

Customization and Tuning

- [z/VM: CP Exit Customization](#), SC24-6269
- [z/VM: Performance](#), SC24-6301

Operation and Use

- [z/VM: CMS Commands and Utilities Reference](#), SC24-6260
- [z/VM: CMS Primer](#), SC24-6265
- [z/VM: CMS User's Guide](#), SC24-6266
- [z/VM: CP Commands and Utilities Reference](#), SC24-6268

- [z/VM: System Operation](#), SC24-6326
- [z/VM: Virtual Machine Operation](#), SC24-6334
- [z/VM: XEDIT Commands and Macros Reference](#), SC24-6337
- [z/VM: XEDIT User's Guide](#), SC24-6338

Application Programming

- [z/VM: CMS Application Development Guide](#), SC24-6256
- [z/VM: CMS Application Development Guide for Assembler](#), SC24-6257
- [z/VM: CMS Application Multitasking](#), SC24-6258
- [z/VM: CMS Callable Services Reference](#), SC24-6259
- [z/VM: CMS Macros and Functions Reference](#), SC24-6262
- [z/VM: CMS Pipelines User's Guide and Reference](#), SC24-6252
- [z/VM: CP Programming Services](#), SC24-6272
- [z/VM: CPI Communications User's Guide](#), SC24-6273
- [z/VM: ESA/XC Principles of Operation](#), SC24-6285
- [z/VM: Language Environment User's Guide](#), SC24-6293
- [z/VM: OpenExtensions Advanced Application Programming Tools](#), SC24-6295
- [z/VM: OpenExtensions Callable Services Reference](#), SC24-6296
- [z/VM: OpenExtensions Commands Reference](#), SC24-6297
- [z/VM: OpenExtensions POSIX Conformance Document](#), GC24-6298
- [z/VM: OpenExtensions User's Guide](#), SC24-6299
- [z/VM: Program Management Binder for CMS](#), SC24-6304
- [z/VM: Reusable Server Kernel Programmer's Guide and Reference](#), SC24-6313
- [z/VM: REXX/VM Reference](#), SC24-6314
- [z/VM: REXX/VM User's Guide](#), SC24-6315
- [z/VM: Systems Management Application Programming](#), SC24-6327
- [z/VM: z/Architecture Extended Configuration \(z/XC\) Principles of Operation](#), SC27-4940

Diagnosis

- [z/VM: CMS and REXX/VM Messages and Codes](#), GC24-6255
- [z/VM: CP Messages and Codes](#), GC24-6270
- [z/VM: Diagnosis Guide](#), GC24-6280
- [z/VM: Dump Viewing Facility](#), GC24-6284
- [z/VM: Other Components Messages and Codes](#), GC24-6300
- [z/VM: VM Dump Tool](#), GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- [z/VM: DFSMS/VM Customization](#), SC24-6274
- [z/VM: DFSMS/VM Diagnosis Guide](#), GC24-6275
- [z/VM: DFSMS/VM Messages and Codes](#), GC24-6276
- [z/VM: DFSMS/VM Planning Guide](#), SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (<https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support>), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/iaa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See *z/VM Performance Data Pump*.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See *Data Pump Messages*.

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc1000_v2r5.pdf), GC35-0151

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (<https://www.ibm.com/docs/en/zos>)

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM 7.3 library, that you may find helpful.

- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *TCP/IP Illustrated, Volume 1: The Protocols*, SR28-5586
- *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture*, SC31-6144
- *Internetworking With TCP/IP Volume II: Implementation and Internals*, SC31-6145
- *Internetworking With TCP/IP Volume III: Client-Server Programming and Applications*, SC31-6146
- *DNS and BIND in a Nutshell*, SR28-4970
- "MIB II Extends SNMP Interoperability," C. Vanderberg, *Data Communications*, October 1990.
- "Network Management and the Design of SNMP," J.D. Case, J.R. Davin, M.S. Fedor, M.L. Schoffstall.
- "Network Management of TCP/IP Networks: Present and Future," A. Ben-Artzi, A. Chandna, V. 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.

Index

Special Characters

(file-based)
 setting up GDBM [112](#)

Numerics

5VMTCP30 virtual machine [33](#), [679](#)
7-bit ASCII [129](#)
8BITIME statement [406](#)

A

abandon request [164](#)
abbreviations and acronyms [721](#)
ACBPOOLSIZE statement [527](#)
ACCEPT_RIP_ROUTE statement [233](#)
access protection [158](#)
aclSourceCacheSize [132](#)
activity log mergedRecord field descriptions [711](#)
activity log records list [707](#)
activity logging [163](#)
adding servers and classes [44](#)
ADDRESSTRANSLATIONPOOLSIZE statement [528](#)
adminDN option [85](#), [131](#), [132](#)
administrator
 configuring DN of [85](#)
 password, specifying [132](#)
 specifying DN of [132](#)
adminPW option [85](#), [131](#), [132](#)
AES encryption method [151](#), [154](#)
allowAnonymousBinds [133](#)
ALTCPHOSTNAME statement [381](#)
alternate server
 specifying [133](#)
ALTRSCSDOMAIN statement [381](#)
altServer option [133](#)
AREA statement [218](#)
ARPAGE statement [529](#)
AS (autonomous system) [194](#)
AS_BOUNDARY_ROUTING statement [219](#)
ASCII-to-EBCDIC table [672](#)
ASCII, 7-bit [129](#)
ASSORTEDPARMS statement [529](#)
ATSIGN statement [14](#)
attribute names in add request records [164](#)
attribute names in modify request records [165](#)
attrOverflowCount [133](#)
audit [134](#)
authenticateOnly server control [138](#)
authentication
 server [118](#), [156](#)
authorization
 native authentication [88](#)
AUTOLOG list [534](#)
AUTOLOG statement [55](#), [349](#), [375](#), [534](#)

B

backbone routes [230](#), [249](#)
backend
 database definitions [127](#)
 options for [132](#)
BADSPoolFILEID statement [381](#)
BITNET domain name [395](#)
BLOCK statement [535](#)
blocked
 connection [134](#)
blockedConnection [134](#)
blockedConnectionTimeout option [134](#)
buffers, data
 ACBPOOLSIZE statement [527](#)
 ADDRESSTRANSLATIONPOOLSIZE statement [528](#)
 CCBPOOLSIZE statement [536](#)
 DATABUFFERLIMITS statement [537](#)
 DATABUFFERPOOLSIZE statement [537](#)
 ENVELOPEPOOLSIZE statement [564](#)
 IPROUTEPOOLSIZE statement [590](#)
 LARGEENVELOPEPOOLSIZE statement [591](#)
 NCBPOOLSIZE statement [595](#)
 RCBPOOLSIZE statement [608](#)
 SCBPOOLSIZE statement [613](#)
 SKCBPOOLSIZE statement [614](#)
 SMALLDATABUFFERPOOLSIZE statement [615](#)
 TCBPOOLSIZE statement [620](#)
 TINYDATABUFFERPOOLSIZE statement [621](#)
 UCBPOOLSIZE statement [627](#)
 VSWITCH CONTROLLER statement [628](#)

C

CCBPOOLSIZE statement [536](#)
CDBM backend
 verifying [109](#)
CDBM configuration entries [99](#)
CDP support
 enabling [458](#)
certificate
 authenticating [156](#)
 digital
 server [118](#)
certificate revocation checking, partner
 SSL/TLS [457](#)
change
 logging [135](#)
 maxage [135](#)
changelog
 logging [135](#)
 LoggingParticipant [135](#)
 maxage [135](#)
 maxentries [135](#)
changeLogging option [135](#)
changeLoggingParticipant [135](#)
changeLogmaxAge [135](#)

- changeLogMaxEntries [135](#)
- character sets [667](#)
- checklist for configuration file [130](#)
- CHECKSUM statement [677](#)
- Checksum testing [677](#)
- CHKIPADR exec [70](#)
- ciphers
 - specifications [157](#)
- classes, server [44](#)
- clear text password [121](#), [151](#), [172](#)
- Client connections [170](#)
- cn=admingroup,cn=configuration [104](#)
- cn=configuration [100](#)
- cn=ibmpolicies [104](#)
- cn=Log Management,cn=Configuration [103](#)
- cn=pwdpolicy,cn=ibmpolicies
 - cn=pwdpolicy [104](#)
- cn=Replication,cn=configuration [101](#)
- cn=Replication,cn=Log Management,cn=Configuration [103](#)
- cn=safadmingroup,cn=configuration [104](#)
- code page IBM-1047 [127](#)
- code pages [667](#)
- command syntax
 - LDAPSRV [80](#)
 - PORTMAP [348](#)
 - REXEC [350](#)
 - SMTP [376](#)
 - SQESERV [447](#)
 - SRVRFTP [56](#)
 - VMNFS [326](#)
 - VMSSL [470](#)
- commands
 - SMSG FTPSERVE, privileged user [74](#)
 - SSL administration [490](#), [497](#), [500](#), [502](#)
- commitCheckpointEntries [135](#)
- commitCheckpointTOD [136](#)
- common configuration statements for RIP and OSPF [257](#)
- commThreads option [136](#)
- communication thread pool [136](#)
- COMPARISON statement [221](#)
- configuration
 - specifying value for distinguished name [129](#)
 - specifying value for filename [129](#)
- configuration file
 - administrator DN, specifying [85](#)
 - alternate [84](#)
 - copying [83](#)
 - creating [127](#)
 - DS CONF [84](#)
 - format [127](#)
 - locating [84](#)
 - options [132](#)
 - options checklist [130](#)
 - password, specifying [85](#)
 - scenarios [125](#)
 - using to configure [84](#)
- configuration file options
 - dnCacheSize [137](#)
- configuration files
 - DTCPARMS [35](#)
 - ETC HOSTS [27](#)
 - IBM DTCPARMS [35](#)
 - local site table [27](#), [28](#)
 - minidisk location summary [50](#), [51](#)

- configuration files (*continued*)
 - MPRoute [203](#)
 - overview [50](#)
 - PW SRC [442](#)
 - RSCSLPD CONFIG [363](#)
 - RSCSLPDTCP CONFIG [361](#)
 - RSCSLPR CONFIG [355](#)
 - RSCSLPRP CONFIG [359](#)
 - RSCSTCP CONFIG [353](#), [663](#)
 - RSCSUFT CONFIG [664](#)
 - SMTP CONFIG [377](#)
 - SMTP RULES [413](#)
 - TCPIP DATA [13](#)
 - UFTD CONFIG [649](#)
 - VMNFS CONFIG [327](#), [334](#)
- configuration statements
 - IPv6 OSPF [241](#)
 - IPv6 RIP OSPF [250](#)
 - OSPF [218](#)
 - RIP [232](#)
- Configure Automatic File Translation Support (FTP) [68](#)
- configuring
 - ETC HOSTS file [27](#)
 - FTP server [55](#)
 - LDBM backend [126](#)
 - local site table [27](#)
 - MPRoute [200](#)
 - non-postscript printer [354](#)
 - PORTMAP server [347](#)
 - postscript printer [356](#)
 - REXEC server [349](#)
 - RSCS
 - LPD link [361](#)
 - LPR link [353](#)
 - TN3270E printer link [365](#)
 - UFT client [663](#)
 - running with SDBM [111](#)
 - scenarios [125](#)
 - servers [35](#)
 - SLAPD [84](#)
 - SMTP server [375](#)
 - SNMP servers [439](#)
 - SSL server [453](#)
 - system parameters [13](#)
 - TCP-to-RSCS Mail Gateway [407](#)
 - TCP-to-RSCS Secure Mail Gateway [409](#)
 - TCPIP DATA file [13](#)
 - TCPIP virtual machine [507](#)
 - Telnet server [584](#)
 - TN3270E printer [353](#)
 - UFT client [663](#)
 - UFTD server [649](#)
 - VMNFS server [325](#)
- Configuring for user password encryption or hashing [123](#)
- configuring the activity log [165](#)
- connections
 - specifying number of [146](#)
 - specifying timeout [139](#)
- copying
 - configuration files [83](#)
- Copying an LDBM database
 - LDBM database [88](#)
- CP directory [33](#)
- CP SMSG Command [342](#)

- creating
 - DS CONF [84](#)
- crypt encryption method [150](#)
- cryptographic mode
 - configuration [463](#)
 - requirements [463](#)
- CSMSERVE virtual machine [34](#)
- CTC DEVICE and LINK statement [539](#)
- customizing
 - DBCS translation tables [673](#)
 - SBCS translation tables [672](#)
 - servers [43](#)
 - SMTP headers [412](#)

D

- data buffers
 - ACBPOOLSIZE statement [527](#)
 - ADDRESSTRANSLATIONPOOLSIZE statement [528](#)
 - CCBPOOLSIZE statement [536](#)
 - DATABUFFERLIMITS statement [537](#)
 - DATABUFFERPOOLSIZE statement [537](#)
 - ENVELOPEPOOLSIZE statement [564](#)
 - IPROUTEPOOLSIZE statement [590](#)
 - LARGEENVELOPEPOOLSIZE statement [591](#)
 - NCBPOOLSIZE statement [595](#)
 - RCBPOOLSIZE statement [608](#)
 - SCBPOOLSIZE statement [613](#)
 - SKCBPOOLSIZE statement [614](#)
 - SMALLDATABUFFERPOOLSIZE statement [615](#)
 - TCBPOOLSIZE statement [620](#)
 - TINYDATABUFFERPOOLSIZE statement [621](#)
 - UCBPOOLSIZE statement [627](#)
 - VSWITCH CONTROLLER statement [628](#)
- database option [136](#)
- databaseDirectory option [137](#)
- DATABUFFERLIMITS statement [537](#)
- DATABUFFERPOOLSIZE statement [537](#)
- DB2PWDEN
 - description [172](#)
- DB2PWDEN utility
 - description [172](#)
- DBCS statement [381](#)
- debug levels [107](#), [163](#)
- debugging facility
 - turning on and off [163](#)
- default
 - referral [153](#)
- DEFAULT_ROUTE statement [257](#)
- defining
 - half-open connections allowed [600](#)
 - MIB data file [441](#), [448](#)
 - nicknames and mailing lists [411](#)
 - participation in native authentication [89](#)
- definitions, virtual machines
 - 5VMTCP30 [33](#)
 - CSMSERVE [34](#)
 - FTPserve [34](#)
 - MROUTE [34](#)
 - PORTMAP [34](#)
 - REXECD [34](#)
 - SMTP [34](#)
 - SNMPD [34](#)
 - SNMPQE [34](#)

- definitions, virtual machines (*continued*)
 - TCPIP [34](#)
 - TCPMAINT [33](#)
 - UFTD [35](#)
 - VMNFS [35](#)
- DEMAND_CIRCUIT statement [221](#)
- DES encryption method [151](#), [154](#)
- DEVICE and LINK statements [539](#)
- devices, network
 - attaching to TCPIP virtual machine [519](#)
- digestRealm option [137](#)
- digital certificate [118](#)
- directory
 - of DS CONF file [84](#)
- DISK option [387](#)
- distinguished name (DN)
 - administrator [85](#), [132](#)
 - master server [145](#)
- distribution tape [509](#)
- Domain Name Resolver [392](#)
- DOMAINLOOKUP statement [14](#)
- DOMAINORIGIN statement [15](#)
- DOMAINSEARCH statement [16](#)
- DS CONF
 - creating [84](#)
 - format [127](#)
 - locating [84](#)
 - See configuration file [128](#)
- DS ENVVARS file [190](#)
- DS2LDIF
 - description [175](#)
- DS2LDIF utility
 - description [175](#)
- DTCPARMS file
 - configuring [36](#)
- DTCPARMS files
 - configuring [36](#)
 - descriptions [35](#)
 - external security manager entries [687](#)
 - search order [35](#)
 - tags and their descriptions [37–43](#)
 - updating for servers
 - FTPserve [55](#)
 - MROUTE [201](#)
 - PORTMAP [347](#)
 - REXECD [349](#)
 - SMTP [375](#)
 - SNMPQE [440](#)
 - TCPIP [507](#), [508](#)
 - UFTD [649](#)
 - VMNFS [325](#)
- DUMPMOUNT statement [328](#)
- duplicating servers [44](#)
- dynamic debugging
 - using [163](#)
- dynamic routing
 - IPv6 [196](#)
 - using MPRoute [194](#)

E

- EARN domain name [395](#)
- EARNET domain name [395](#)
- EBCDIC-to-ASCII table [672](#)

- enableResources option [137](#)
- enabling
 - SSL [117](#)
- encryption
 - configuring for [121](#)
 - for communication channel [118](#)
- encryption, password
 - See password encryption [150](#)
- ENVELOPEPOOLSIZE statement [564](#)
- environment variables
 - LANG [190](#)
 - NLSPATH [190](#)
- Environment variables
 - variables used by the LDAP server [105](#)
- escape mechanism for UTF-8 characters [129](#)
- ESM (External Security Manager)
 - FTP server [57](#)
 - LDAPSRV change logging [127](#)
 - NFS server [327](#)
 - REXEC server [351](#)
 - using TCP/IP with [685](#), [690](#)
- ETC HOSTS file [27](#)
- examples
 - configuration scenarios [125](#)
 - DS ENVVARS file [190](#)
 - LDAP URL [140](#)
 - setting up native authentication [95](#)
- execs
 - TCPCOMP [682](#)
 - TCPLOAD [681](#)
 - TCPTXT [681](#)
 - VMFC [680](#)
 - VMFPAS [679](#)
- EXPORT statement [329](#)
- EXPORTONLY statement [330](#)
- extended group membership searching [138](#)
- extendedGroupSearching option [138](#)
- External Security Manager (ESM)
 - FTP server [57](#)
 - NFS server [327](#)
 - REXEC server [351](#)
 - using TCP/IP with [685](#), [690](#)

F

- FILE statement [565](#)
- File system [171](#)
- File Transfer Protocol (FTP) [55](#)
- files
 - configuration, global options [132](#)
 - copying configuration [83](#)
 - DS CONF [128](#)
 - stash [158](#)
- fileTerminate [138](#)
- FILTER statement [233](#)
- filterCacheBypassLimit [138](#)
- filterCacheSize [138](#)
- Finalizing setup
 - setup of LDAP backends [110](#)
- FINISHOPEN statement [383](#)
- FIXEDPAGESTORAGEPOOL statement [566](#)
- FOREIGNIPCONLIMIT statement [567](#)
- formats
 - DS CONF [128](#)

- FORWARDMAIL statement [383](#)
- free pool statements [521](#)
- FTP
 - server [55](#)
 - welcome banner [69](#), [71](#)
- FTP server exit [70](#)
- FTPKEEPALIVE statement [60](#)
- FTPPERM command [689](#)
- FTPSEVER virtual machine [34](#)
- functions
 - application programming interface [3](#)
 - client [3](#)
 - connectivity and gateway [1](#)
 - network status/management [3](#)
 - server [1](#)
- futile neighbor state loops [199](#)

G

- GATEWAY statement [384](#), [568](#)
- gathering group memberships [138](#)
- GCS servers [49](#)
- GDBM (file-based) backend-specific section
 - section in DS CONF [128](#)
- GDBM backend
 - setting up for [112](#)
 - verifying [109](#)
- GetEffectiveACL [187](#)
- global configuration file options [132](#)
- global section in DS CONF [127](#)
- Global server exit [48](#)
- GLOBAL_OPTIONS statement [259](#)
- groups
 - extended, membership searching [138](#)

H

- half-open connections, defining [600](#)
- HiperSockets DEVICE and LINK statement [542](#)
- HOME list [579](#)
- HOME statement [579](#)
- HOMETEST [25](#)
- HOST statement [28](#)
- HOSTNAME statement [18](#)
- HOSTS ADDRINFO file [30](#)
- HOSTVERIFICATION statement [18](#)
- HTTP CDP support
 - enabling [458](#)

I

- IBM DTCPARMS file [9](#), [35](#), [50](#)
- ICHRXC02 exit [688](#)
- ICMPERRORLIMIT statement [583](#)
- IDENTIFY command [657](#)
- IDENTIFY statement [650](#)
- idleConnectionTimeout option [139](#)
- IFCONFIG command [631](#)
- IGNORE_RIP_NEIGHBOR statement [234](#)
- INACTIVE statement [385](#)
- include option [139](#)
- INCLUDE statement [204](#)
- INFORM statement [584](#)

- initializing
 - native authentication [88](#)
- installing
 - RACF [111](#)
- INTERFACE statement [258](#)
- INTERNALCLIENTPARMS statement [584](#)
- IPCONFIG
 - MULTIPATH [195](#)
- IPMAILERADDRESS statement [385](#)
- IPROUTEPOOLSIZE statement [590](#)
- IPv6
 - dynamic routing [196](#)
 - IPv6_ACCEPT_RIP_ROUTE statement [250](#)
 - IPv6_AREA statement [241](#)
 - IPv6_AS_BOUNDARY_ROUTING statement [242](#)
 - IPv6_IGNORE_RIP_NEIGHBOR statement [251](#)
 - IPv6_ORIGINATE_RIP_DEFAULT statement [252](#)
 - IPv6_OSPF statement [244](#)
 - IPv6_OSPF_INTERFACE statement [245](#)
 - IPv6_RANGE statement [248](#)
 - IPv6_RIP_FILTER statement [251](#)
 - IPv6_RIP_INTERFACE statement [253](#)
 - IPv6_RIP_SEND_ONLY statement [256](#)
 - IPv6_VIRTUAL_LINK statement [249](#)
 - OSPF configuration statements [241](#)
 - RIP configuration statements [250](#)
- IPv6_DEFAULT_ROUTE statement [259](#)
- IPv6_INTERFACE statement [260](#)
- IUCV connections DEVICE and LINK statement
 - local [545](#)
 - remote [548](#)

J

- Japanese messages [190](#)

K

- KEEPALIVEOPTIONS statement [590](#)
- key database file
 - specifying for server [158](#)

L

- LANG environment variable [190](#)
- LARGEENVELOPEPOOLSIZE statement [591](#)
- LCS DEVICE and LINK statement [551](#)
- LDAP 2
 - output data format [154](#)
 - validating data [161](#)
- LDAP 3
 - UTF-8 characters [191](#)
- LDAP backend utilities
 - running and using [171](#)
- LDAP client
 - cipher specifications [157](#)
- LDAP client requests
 - listening for [139](#)
- LDAP server
 - administrator [132](#)
 - administrator distinguished name (DN) [85](#)
 - alternate server [133](#)
 - configuration options [127](#)

- LDAP server (*continued*)
 - configuring [84](#)
 - cryptographic hardware, using [115](#)
 - debugging facility [163](#)
 - defining user ID [82](#)
 - enabling SSL for [117](#)
 - equivalent server [133](#)
 - extended group membership searching [138](#)
 - multiple single-server mode LDAP servers [84](#)
 - NLS [190](#)
 - preparing [82](#)
 - RACF resources and classes, operations on [137](#)
 - requirements for user ID [82](#)
 - setting up LDBM [87](#)
 - setting up SDBM [111](#)
 - single-server mode [84](#)
 - updating DTCPARMS [80](#)
 - updating PROFILE TCPIP [79](#)
 - verifying [109](#)
- LDAP URL
 - examples [140](#)
- LDAP URL format
 - specifying for listen [139](#)
- LDAP_DEBUG environment variable [109](#)
- ldap_ssl_client_init API
 - using for SASL bind [121](#)
- ldap_ssl_init API
 - using for SASL bind [121](#)
- LDAPEXOP utility [181](#)
- LDAPSrch
 - using to verify LDAP server [109](#)
- LDAPSrch utility
 - using to verify LDAP server [109](#)
- LDAPSRV command syntax [80](#)
- LDBM backend
 - section in DS CONF [127](#)
 - setting up for [87](#)
- LESSTRACE statement [592](#)
- levels, debug [107](#)
- limit, time
 - specifying in configuration file [159](#)
- listen option [139](#)
- load list [682](#)
- local site table 27, 50
- local site table configuration statements
 - HOST [28](#)
 - NET [29](#)
- LOCALFORMAT statement [386](#)
- locating
 - DS CONF [84](#)
- LOG statement [387](#)
- logfile option [142](#)
- logFileFilter option [142](#)
- logFileMicroseconds option [142](#)
- logFileMsgs option [143](#)
- logFileOps option [143](#)
- logFileRecordType option [143](#)
- logFileRolloverDirectory option [144](#)
- logFileRolloverSize option [144](#)
- logFileRolloverTOD option [144](#)
- logFileVersion option [145](#)
- Logging
 - Participant [135](#)
- logging requests unknown to activity logging [165](#)

loopback testing [677](#)

M

- mail gateway [407](#)
- MAILER statement [387](#)
- MAILHOPCOUNT statement [388](#)
- MAKESITE program [30](#)
- making changes to the site table [30](#)
- managing password
 - managing administrator DN [85](#)
- mask for debug level [107](#)
- master
 - server DN [145](#)
 - server password [146](#)
 - server, specifying [145](#)
- masterServer option [145](#)
- masterServerDN option [145](#)
- masterServerPW option [146](#)
- maxConnections option [146](#)
- MAXFILEBYTES statement [651](#)
- MAXMAILBYTES statement [389](#)
- MAXTCPUSERS statement [330](#)
- maxThreads option [136](#)
- MD5 and OSPF [202](#), [219](#)
- MD5 encryption method [151](#)
- membership
 - extended group, searching [138](#)
- mergedRecord activity log fields [711–714](#)
- mergedRecords
 - activity log start and end field descriptions
 - activity log field descriptions [707](#)
- message examples, notation used in [xxiv](#)
- migration [4](#)
- modes
 - single-server mode [84](#)
 - single-server mode LDAP servers [84](#)
- Monitoring LDAP server resources
 - server resources [170](#)
- MONITORRECORDS statement [593](#)
- MORETRACE statement [595](#)
- MPRoute
 - ACCEPT_RIP_ROUTE statement [233](#)
 - AREA statement [218](#)
 - Areas [225](#), [247](#)
 - AS_BOUNDARY_ROUTING statement [219](#)
 - authentication [232](#), [240](#)
 - Authentication_Type [219](#)
 - backbone routes [230](#)
 - common configuration statements for RIP and OSPF [257](#)
 - COMPARISON statement [221](#)
 - configuration file [203](#)
 - configuring server [193](#)
 - DEFAULT_ROUTE statement [257](#)
 - DEMAND_CIRCUIT statement [221](#)
 - designed router [228](#)
 - DR_Max_Adj_Attempt [222](#), [245](#)
 - FILTER statement [233](#)
 - futile neighbor state loops [199](#)
 - GLOBAL_OPTIONS statement [259](#)
 - IGNORE_RIP_NEIGHBOR statement [234](#)
 - importing routes to OSPF [219](#), [242](#)
 - INCLUDE statement [204](#)

MPRoute (continued)

- interaction with VIPA [199](#)
- INTERFACE statement [258](#)
- interfaces [221](#), [223](#), [244](#), [245](#)
- IPv6 OSPF configuration [241](#)
- IPv6 RIP configuration [250](#)
- IPv6_ACCEPT_RIP_ROUTE statement [250](#)
- IPv6_AREA statement [241](#)
- IPv6_AS_BOUNDARY_ROUTING statement [242](#)
- IPv6_DEFAULT_ROUTE statement [259](#)
- IPv6_FILTER statement [251](#)
- IPv6_IGNORE_RIP_NEIGHBOR statement [251](#)
- IPv6_INTERFACE statement [260](#)
- IPv6_ORIGINATE_RIP_DEFAULT statement [252](#)
- IPv6_OSPF areas [248](#)
- IPv6_OSPF statement [244](#)
- IPv6_OSPF_INTERFACE statement [245](#)
- IPv6_RANGE statement [248](#)
- IPv6_RIP_INTERFACE statement [253](#)
- IPv6_RIP_SEND_ONLY statement [256](#)
- IPv6_VIRTUAL_LINK statement [249](#)
- Link State Advertisements (LSAs) [228](#), [248](#)
- metrics [221](#)
- ORIGINATE_RIP_DEFAULT statement [234](#)
- OSPF areas [229](#)
- OSPF configuration statements [218](#)
- OSPF statement [221](#)
- OSPF_INTERFACE statement [223](#)
- overview [194](#)
- RANGE statement [229](#)
- RIP configuration [232](#)
- RIP_INTERFACE statement [235](#)
- RouterID statement [230](#)
- security [219](#), [232](#), [240](#)
- SEND_ONLY statement [240](#)
- SMSG Interface [161](#), [261](#)
- stopping [264](#), [322](#)
- stub area [219](#), [242](#)
- supported protocols [194](#)
- syntax rules [204](#)
- VIRTUAL_LINK statement [230](#)

MROUTE Command [203](#)

MROUTE CONFIG file [201](#)

MULTIPATH (IPCONFIG MULTIPATH) [195](#)

multiple interface network support [512](#)

multiple SSL server configuration [462](#)

Multiprocessor Configuration [507](#)

MX records [377](#)

N

- National Language Support (NLS)
 - setting variables for [190](#)
- native authentication
 - defining participation [89](#)
 - description [88](#)
 - enabling [161](#)
 - example of setting up [95](#)
 - initializing [88](#)
 - installing RACF for [111](#)
 - operating modes [89](#)
 - password changes [147](#)
 - specifying DN [147](#)
 - updating schema [88](#)

- native authentication (*continued*)
 - using with Web servers [99](#)
- native operations
 - running [91](#)
- nativeAuthSubtree option [89](#), [147](#)
- nativeUpdateAllowed option [89](#), [147](#)
- NCBPOOLSIZE statement [595](#)
- NETDATA format [387](#), [396](#)
- NETNORTH domain name [395](#)
- NETSTAT CP command [515](#), [521](#), [528](#)
- network attachments [11](#)
- Network communications [170](#)
- network devices
 - attaching to TCPIP virtual machine [519](#)
- NFS CMS MSG Command [342](#), [344](#)
- NFSPERM command [689](#)
- NLSPATH environment variable [190](#)
- NOCHECKSUM statement [677](#)
- NOHEADER option [386](#)
- NOLOG statement [389](#)
- NOSCREEN statement [596](#)
- notation used in message and response examples [xxiv](#)
- NOTRACE statement [597](#)
- NSINTERADDR statement [19](#)
- NSLOOKUP
 - command [658](#)
 - exit [655](#)
 - statement [651](#)
- NSPORTADDR statement [20](#)

O

- obey list [597](#)
- OBEY statement [597](#)
- OBEYFILE command [644](#)
- OCSPParms tag [459](#)
- ONDISKFULL statement [390](#)
- One-way hashing formats [122](#)
- operating modes
 - native authentication [89](#)
- operationsMonitor
 - monitor [147](#)
- operationsMonitor option [147](#)
- operationsMonitorSize
 - monitor [148](#)
- operationsMonitorSize option [148](#)
- options
 - checklist [130](#)
 - configuration file [132](#)
- ORIGINATE_RIP_DEFAULT statement [234](#)
- OSA-Express adapter support [518](#)
- OSCP support
 - enabling [457](#)
- OSPF
 - common configuration statements [257](#)
 - configuration statements [218](#)
 - DEFAULT_ROUTE statement [257](#)
 - GLOBAL_OPTIONS statement [259](#)
 - hierarchy [221](#)
 - INTERFACE statement [258](#)
 - IPv6_DEFAULT_ROUTE statement [259](#)
 - IPv6_INTERFACE statement [260](#)
 - IPv6_OSPF statement [244](#)
 - IPv6_OSPF_INTERFACE statement [245](#)

- OSPF (*continued*)
 - OSPF statement [221](#)
 - OSPF_INTERFACE statement [223](#)
- OSPF (open shortest path first)
 - configuring authentication [202](#)
 - configuring OSPF and RIP [205](#)
 - overview [194](#)
- OSPF MD5 Authentication [282](#)
- OSPF statement [221](#)
- OSPF_INTERFACE statement [223](#)
- OUTBOUNDOPENLIMIT statement [391](#)

P

- PACKETTRACESIZE statement [598](#), [600](#)
- parameter, Subnet_mask [208](#)
- partner certificate revocation checking
 - SSL/TLS [457](#)
- password
 - administrator, specifying [132](#)
 - changing [147](#)
 - master server [146](#)
 - SSL key database file [158](#)
 - storing in stash file [158](#)
- password encryption
 - DB2PW DEN utility [172](#)
 - pwEncryption configuration option [150](#)
- password or password phrases
 - native modify [91](#)
- Password policy with native authentication [95](#)
- PC (program call)
 - See program call (PC) [148](#)
- pcIdleConnectionTimeout option [148](#)
- PCNFSD statement [331](#)
- pcThreads option [148](#)
- peerServerdn [148](#)
- peerServerPW [149](#)
- PENDINGCONNECTIONLIMIT statement [600](#)
- PERMIT statement [601](#)
- persistentSearch [149](#)
- plug-in
 - pluginParameters [150](#)
- PORT statement
 - SMTP server [391](#)
 - TCPIP server [603](#)
 - UFTD server [652](#)
- PORTMAP command syntax [348](#)
- PORTMAP virtual machine [34](#)
- portmapper service, verifying [348](#)
- POSTMASTER statement [391](#)
- preparing
 - LDAP server [82](#)
- PRIMARYINTERFACE statement [606](#)
- privileged user SMSG commands [420](#), [426](#)
- PRMODE operand on CP TAG command [373](#)
- PROFILE EXEC [46](#)
- program call (PC)
 - initializing threads for [148](#)
 - pcIdleConnectionTimeout [148](#)
- protecting access [158](#)
- PUNCH format [387](#), [396](#)
- PW SRC file [442](#)
- pwCryptCompat [150](#)
- pwEncryption option [150](#), [172](#)

pwSearchOutput [152](#)

Q

QUERY command [658](#)

QUIT command [659](#)

R

RACF (Resource Access Control Facility)

how to use [57](#), [351](#)

security requirements [685](#)

RACF resources and classes, operations on [137](#)

RANGE statement [229](#)

RCBPOOLSIZE statement [608](#)

RCPT commands [392](#)

RCPTRESPONSEDELAY statement [392](#)

RDBM backend

verifying [109](#)

readOnly option [152](#)

referral option [153](#)

referrals

default [153](#)

specifying [153](#)

related protocols [715](#)

Remote Execution Command Protocol (REXEC) [349](#)

Remote Spooling Communications Subsystem (RSCS)

configuring

LPD link [361](#)

LPR link [353](#)

TN3270E printer link [365](#)

domain names [395](#)

network [385](#), [395](#), [407](#)

replaceable static route

smsg MPRoute [292](#)

replica

master server [145](#)

RESOLVERRETRYINT statement [393](#)

RESOLVERTIMEOUT statement [20](#)

RESOLVERUDPRETRIES statement [20](#)

RESOLVEVIA statement [21](#)

Resource Access Control Facility (RACF)

accessing information in [87](#), [111](#), [112](#)

administrator DN [86](#)

installing for native authentication [111](#)

installing for SDBM [111](#)

password [86](#)

resources and classes, RACF [137](#)

response examples, notation used in [xxiv](#)

RESTRICT statement [393](#), [608](#)

RETRYAGE statement [394](#), [401](#)

RETRYINT statement [394](#)

revocation checking, partner certificate

SSL/TLS [457](#)

REWRITE822HEADER statement [394](#)

REXECD

server, starting [350](#)

statement [350](#)

virtual machine [34](#), [349](#)

REXECD command syntax [350](#)

RIP

ACCEPT_RIP_ROUTE statement [233](#)

AS boundary routing capability [219](#)

RIP (*continued*)

common configuration statements for RIP and OSPF [257](#)

configuration statements [232](#)

DEFAULT_ROUTE statement [257](#)

FILTER statement [233](#)

GLOBAL_OPTIONS statement [259](#)

IGNORE_RIP_NEIGHBOR statement [234](#)

INTERFACE statement [258](#)

IPv6_ACCEPT_RIP_ROUTE statement [250](#), [251](#)

IPv6_DEFAULT_ROUTE statement [259](#)

IPv6_FILTER statement [251](#)

IPv6_IGNORE_RIP_NEIGHBOR statement [251](#)

IPv6_INTERFACE statement [260](#)

IPv6_ORIGINATE_RIP_DEFAULT statement [252](#)

IPv6_RIP_INTERFACE statement [253](#)

IPv6_RIP_SEND_ONLY statement [256](#)

ORIGINATE_RIP_DEFAULT statement [234](#)

RIP_INTERFACE statement [235](#)

SEND_ONLY statement [240](#)

RIP (Routing Information Protocol)

configuring [205](#)

definition [195](#)

RIP interface, RIP1 and RIP2 packets [288](#)

RIP_INTERFACE statement [235](#)

ROUTERADV statement [609](#), [611](#)

RouterID statement [230](#)

routing

IPv6 dynamic [196](#)

MULTIPATH [195](#)

server [194](#)

Routing Information Protocol statements, also see RIP [232](#)

RSCS (Remote Spooling Communications Subsystem)

configuring

LPD link [361](#)

LPR link [353](#)

TN3270E printer link [365](#)

domain names [395](#)

network [385](#), [395](#), [407](#)

RSCS UFT client [663](#)

RSCSDOMAIN statement [395](#)

RSCSFORMAT statement [395](#)

RSCSLPD CONFIG file [363](#)

RSCSLPDTCP CONFIG file [361](#)

RSCSLPR CONFIG file [355](#)

RSCSLPRP CONFIG file [359](#)

RSCSTCP CONFIG file [353](#), [663](#)

RSCSUFT CONFIG file [664](#)

RSHELL port [350](#)

running

native operations [91](#)

S

sample files, references to

CHKIPADR EXEC [72](#)

local site table [27](#)

MIB_DESC DATA [448](#)

MPROUTE SCONFIG [50](#)

PROFILE TCPIP [509](#)

PW SRC [443](#)

SECURITY MEMO [410](#)

SMTP CONFIG [377](#)

SMTP SECTABLE [409](#)

sample files, references to (*continued*)

TCPIP DATA [13](#)

samples

DS ENVVARS file [190](#)

SCBPOOLSIZE statement [613](#)

scenarios

configuration [125](#)

schema

updating for native authentication [88](#)

schemaPath [153](#)

schemaReplaceByValue [153](#)

SCREEN statement [614](#)

SDBM backend

installing RACF for [111](#)

section in DS CONF [127](#)

setting up for [111](#)

verifying [109](#)

secondary machines [351](#)

secretEncryption option [154](#)

secure ports, defining [484](#)

Secure Socket Layer (SSL)

commands [490](#), [497](#), [500](#), [502](#)

configuring [453](#)

dynamic server operation [485](#)

overview of SSL session [454](#)

secure ports, defining [484](#)

server

starting [485](#)

stopping [486](#)

SSLIDCSS command [467](#)

understanding certification validation [455](#)

Secure Sockets Layer (SSL)

certificate [156](#)

certificate authentication [156](#)

cipher specifications [157](#)

enabling [117](#)

key database file

protecting access to [158](#)

specifying for server [158](#)

password for key database file [158](#)

server

tracing server activities [487](#)

setting up options for [116](#)

stash file [158](#)

SECURE statement [396](#)

SECURETELNETCLIENT statement [21](#)

security

options, setting up [116](#)

security interfaces [685](#)

security option [117](#)

SEND_ONLY statement [240](#)

sendV3stringoverV2as option [154](#)

server

alternate [133](#)

certificate [118](#)

master, specifying [145](#)

Server backends during startup [170](#)

server configuration [35](#)

server controls

authenticateOnly [138](#)

server session cache

monitoring [486](#)

serverEtherAddr option [154](#)

servers

servers (*continued*)

adding [44](#)

customizing [43](#)

duplicating [44](#)

setting time zone

time zone [105](#)

setting up

native authentication example [95](#)

SHA encryption method [151](#)

single SSL server configuration [463](#)

single-server mode

multiple [84](#)

running in [84](#)

sizelimit

specifying in configuration file [155](#)

sizeLimit option [155](#)

SKCBPOOLSIZE statement [614](#)

SMALLDATABUFFERPOOLSIZE statement [615](#)

SMF records

Record Type 83, subtype 3 [691](#)

SMSG (CMS Subcommand) [342](#), [344](#)

SMSG interface [74](#)

SMSG SMTP commands, privileged user

FORWARDMAIL [421](#)

LISTMAIL [423](#)

MAILINFO [425](#)

PURGE [426](#)

REFRESH [427](#)

REPROCESS [428](#)

SMTPCMDS [429](#)

SOURCEROUTES [431](#)

SUPPRESSNOTIFICATION [400](#)

TRACE [435](#)

VERIFYCLIENT [436](#)

SMSGAUTHLIST statement [396](#)

SMTP

CONFIG file [377](#)

local and non-local mail recipients [377](#)

NAMES file [411](#)

security table [409](#)

server [375](#)

SMTPRSCS [408](#)

SMTPRSCS HOSTINFO file [385](#)

virtual machine [34](#)

SMTP command syntax [376](#)

SMTP CONFIG file [377](#)

SMTP configuration statements

8BITMIME [406](#)

ALTRSCSDOMAIN [381](#)

ALTTCPOSTNAME [381](#)

BADSPoolFILEID [381](#)

DBCS [381](#)

FINISHOPEN [383](#)

FORWARDMAIL [383](#)

GATEWAY [384](#)

INACTIVE [385](#)

IPMAILERADDRESS [385](#)

LOCALFORMAT [386](#)

LOG [387](#), [388](#)

MAILER [387](#)

MAXMAILBYTES [389](#)

NOLOG [389](#)

ONDISKFULL [390](#)

OUTBOUNDOPENLIMIT [391](#)

SMTP configuration statements (*continued*)

- PORT [391](#)
- POSTMASTER [391](#)
- RCPTRESPONSEDELAY [392](#)
- RESOLVERRETRYINT [393](#)
- RESTRICT [393](#)
- RETRYAGE [394](#)
- RETRYINT [394](#)
- REWRITE822HEADER [394](#)
- RSCSDOMAIN [395](#)
- RSCSFORMAT [395](#)
- SECURE [396](#)
- SMSGAUTHLIST [396](#)
- SMTPCMDS [397](#)
- SOURCEROUTES [399](#)
- SUPPRESSNOTIFICATION [400](#)
- TEMPERRORRETRIES [400](#)
- TRACE [402](#)
- VERIFYBATCHSMTPSENDER [403](#)
- VERIFYCLIENT [404](#)
- VERIFYCLIENTDELAY [405](#)
- WARNINGAGE [405](#)
- SMTP RULES file [413](#), [419](#)
- SMTPCMDS statement [397](#)
- SMTPSERVERID statement [21](#)
- SNMP
 - SNMP client [445](#)
 - SNMP overview [439](#), [445](#)
- SNMPD virtual machine [34](#)
- SNMPQE virtual machine [34](#)
- SNMPTRAP DEST file [442](#)
- SOMAXCONN statement [616](#)
- source code libraries
 - TCPCOMP [682](#)
 - TCPLOAD [681](#)
 - TCPTCT [681](#)
 - using [679](#)
 - VMFASM [679](#)
 - VMFC [680](#)
 - VMFPAS [679](#)
- SOURCEROUTES statement [399](#)
- SQESERV command syntax [447](#)
- SRVRFTP command syntax [56](#)
- srvStartUpError [156](#)
- SSL (Secure Socket Layer)
 - commands [490](#), [497](#), [500](#), [502](#)
 - configuring [453](#)
 - dynamic server operation [485](#)
 - overview of SSL session [454](#)
 - secure ports, defining [484](#)
 - server
 - starting [485](#)
 - stopping [486](#)
 - SSLIDCSS command [467](#)
 - understanding certification validation [455](#)
- SSL (Secure Sockets Layer)
 - server
 - tracing server activities [487](#)
- SSL DCSS Management agent virtual machine [463](#)
- SSL pool server virtual machine [463](#)
- SSL/TLS
 - creating key database [116](#)
 - cryptographic hardware, using [465](#)
 - enabling [116](#)

SSL/TLS (*continued*)

- LDAP utilities [171](#)
- obtaining a certificate [116](#)
- partner certificate revocation checking [457](#)
- protected communications [114](#)
- setting up an LDAP client [121](#)
- setting up for [114](#)
- SSLADMIN [488](#)
- SSLADMIN CLEAR command [490](#)
- SSLADMIN Cclosecon command [490](#)
- SSLADMIN HELP command [490](#)
- SSLADMIN LOG command [490](#)
- SSLADMIN QUERY command [491](#)
- SSLADMIN REFRESH command [497](#)
- SSLADMIN REStart command [497](#)
- SSLADMIN SET command [497](#)
- SSLADMIN START command [498](#)
- SSLADMIN STOP command [498](#)
- SSLADMIN SYStem command [498](#)
- SSLADMIN TRACE/NOTRACE command [500](#)
- sslAuth option [121](#), [156](#)
- sslCertificate option [156](#)
- sslCipherSpecs option [157](#)
- SSLIDCSS command [467](#)
- sslKeyRingFile option [158](#)
- sslKeyRingFilePW option [158](#)
- sslKeyRingFilePWStashFile option [158](#)
- SSLPOOL command [502](#)
- start and end connection logging [164](#)
- Start or end activity log fields [707–710](#)
- START statement [618](#)
- started task
 - changing debug setting [163](#)
- stash file [158](#)
- statement syntax [524](#)
- statements
 - ACCEPT_RIP_ROUTE [233](#)
 - AREA statement [218](#)
 - AS_BOUNDARY_ROUTING statement [219](#)
 - common configuration statements for RIP and OSPF [257](#)
 - COMPARISON statement [221](#)
 - DEFAULT_ROUTE [257](#)
 - DEMAND_CIRCUIT statement [221](#)
 - FILTER [233](#)
 - GLOBAL_OPTIONS [259](#)
 - IGNORE_RIP_NEIGHBOR [234](#)
 - INCLUDE statement [204](#)
 - INTERFACE [258](#)
 - IPv6 OSPF configuration [241](#)
 - IPv6 RIP configuration [250](#)
 - IPv6_ACCEPT_RIP_ROUTE [250](#)
 - IPv6_AREA statement [241](#)
 - IPv6_AS_BOUNDARY_ROUTING statement [242](#)
 - IPv6_DEFAULT_ROUTE [259](#)
 - IPv6_FILTER [251](#)
 - IPv6_IGNORE_RIP_NEIGHBOR [251](#)
 - IPv6_INTERFACE statement [260](#)
 - IPv6_ORIGINATE_RIP_DEFAULT [252](#)
 - IPv6_OSPF statement [244](#)
 - IPv6_OSPF_INTERFACE statement [245](#)
 - IPv6_RANGE statement [248](#)
 - IPv6_RIP_INTERFACE [253](#)
 - IPv6_RIP_SEND_ONLY [256](#)

statements (*continued*)

- IPv6_VIRTUAL_LINK statement [249](#)
- ORIGINATE_RIP_DEFAULT [234](#)
- OSPF statement [221](#)
- OSPF_INTERFACE statement [223](#)
- RANGE statement [229](#)
- RIP configuration [232](#)
- RIP_INTERFACE [235](#)
- RouterID statement [230](#)
- SEND_ONLY [240](#)
- VIRTUAL_LINK statement [230](#)

statements, local site table configuration file

- HOST [28](#)
- NET [29](#)

statements, SMTP configuration

- 8BITMIME [406](#)
- ALTRCSDOMAIN [381](#)
- ALTTCPHOSTNAME [381](#)
- BADSPoolFILEID [381](#)
- DBCS [381](#)
- FINISHOPEN [383](#)
- FORWARDMAIL [383](#)
- GATEWAY [384](#)
- INACTIVE [385](#)
- IPMAILERADDRESS [385](#)
- LOCALFORMAT [386](#)
- LOG [387](#), [388](#)
- MAILER [387](#)
- MAXMAILBYTES [389](#)
- NOLOG [389](#)
- ONDISKFULL [390](#)
- OUTBOUNDOPENLIMIT [391](#)
- PORT [391](#)
- POSTMASTER [391](#)
- RCPTRESPONSEDELAY [392](#)
- RESOLVERRETRYINT [393](#)
- RESTRICT [393](#)
- RETRYAGE [394](#)
- RETRYINT [394](#)
- REWRITE822HEADER [394](#)
- RSCSDOMAIN [395](#)
- RSCSFORMAT [395](#)
- SECURE [396](#)
- SMSGAUTHLIST [396](#)
- SMTPCMDS [397](#)
- SOURCEROUTES [399](#)
- SUPPRESSNOTIFICATION [400](#)
- TEMPERORRETRIES [400](#)
- TRACE [402](#)
- VERIFYBATCHSMTPSENDER [403](#)
- VERIFYCLIENT [404](#)
- VERIFYCLIENTDELAY [405](#)
- WARNINGAGE [405](#)

statements, TCPIP configuration

- ACBPOOLSIZE [527](#)
- ADDRESSTRANSLATIONPOOLSIZE [528](#)
- ARPAGE [529](#)
- ASSORTEDPARMS [529](#)
- AUTOLOG [534](#)
- BLOCK [535](#)
- CBPOOLSIZE [536](#)
- CTC DEVICE and LINK [539](#)
- DATABUFFERLIMITS [537](#)
- DATABUFFERPOOLSIZE [537](#)

statements, TCPIP configuration (*continued*)

- ENVELOPEPOOLSIZE [564](#)
- FILE [565](#)
- FIXEDPAGESTORAGEPOOL [566](#)
- FOREIGNIPCONLIMIT [567](#)
- GATEWAY [568](#)
- HOME [579](#)
- ICMPERRORLIMIT [583](#)
- INFORM [584](#)
- INTERNALCLIENTPARMS [584](#)
- IPROUTEPOOLSIZE [590](#)
- IUCV connections DEVICE and LINK
 - local [545](#)
 - remote [548](#)
- KEEPALIVEOPTIONS [590](#)
- LAN Channel (LCS) DEVICE and LINK [551](#)
- LARGEENVELOPEPOOLSIZE [591](#)
- LESSTRACE [592](#)
- MONITORRECORDS [593](#)
- MORETRACE [595](#)
- NCBPOOLSIZE [595](#)
- NOSCREEN [596](#)
- NOTRACE [597](#)
- OBEY [597](#)
- OSD DEVICE and LINK [553](#)
- OSH DEVICE and LINK [559](#)
- PACKETTRACE SIZE [598](#), [600](#)
- PENDINGCONNECTIONLIMIT [600](#)
- PERMIT [601](#)
- PORT [603](#), [606](#)
- PRIMARYINTERFACE [606](#)
- RCBPOOLSIZE [608](#)
- RESTRICT [608](#)
- ROUTERADV [609](#)
- ROUTERADVPREFIX [611](#)
- SCBPOOLSIZE [613](#)
- SCREEN [614](#)
- SKCBPOOLSIZE [614](#)
- SMALLDATABUFFERPOOLSIZE [615](#)
- SOMAXCONN [616](#)
- START [618](#)
- STOP [618](#)
- SYSCONTACT [619](#)
- SYSLOCATION [619](#)
- TCBPOOLSIZE [620](#)
- TIMESTAMP [621](#)
- TINYDATABUFFERPOOLSIZE [621](#)
- TN3270E [622](#)
- TRACE [623](#)
- TRACEONLY [625](#)
- TRANSLATE [626](#)
- UCBPOOLSIZE [627](#)
- UDPQUEUELIMIT [627](#)
- VIPA DEVICE and LINK [563](#)
- VSWITCH CONTROLLER [628](#)

statements, TCPIP DATA client configuration

- ATSIGN [14](#)
- DOMAINLOOKUP [14](#)
- DOMAINORIGIN [15](#)
- DOMAINSEARCH [16](#)
- HOSTNAME [18](#)
- HOSTVERIFICATION [18](#)
- NSINTERADDR [19](#)
- NSPORTADDR [20](#)

statements, TCPIP DATA client configuration (*continued*)

- [RESOLVERTIMEOUT 20](#)
- [RESOLVERUDPTRIES 20](#)
- [RESOLVEVIA 21](#)
- [SMTPSERVERID 21](#)
- [TCPIPUSERID 22](#)
- [TRACE RESLOVER 22](#)
- [UFTSERVERID 23](#)
- [USERDATA 23](#)
- [VMFILETYPE 23](#)
- [VMFILETYPEDEFAULT 24](#)

statements, testing

- checksum testing
 - [CHECKSUM 677](#)
 - [NOCHECKSUM 677](#)

statements, UFTD configuration

- [IDENTIFY 650](#)
- [MAXFILEBYTES 651](#)
- [NSLOOKUP 651](#)
- [PORT 652](#)
- [TRACE 652](#)
- [TRANSLATE 653](#)
- [UFTCMDS EXIT 653](#)

STOP command [659](#)

STOP statement [618](#)

Subnet_mask parameter [208](#)

suffix

- option [85](#), [158](#)

SUPPRESSNOTIFICATION statement [400](#)

Symmetric encryption keys [123](#)

syntax

- [LDAPSRV command 80](#)
- [PORTMAP command 348](#)
- [REXECD command 350](#)
- [SMTP command 376](#)
- [SQESERV command 447](#)
- [SRVRFTP command 56](#)
- [VMNFS command 326](#)
- [VMSSL command 470](#)

syntax diagrams, how to read [xxi](#)

SYSCONTACT statement [619](#)

SYSLOCATION statement [619](#)

T

tables

- configuration file summary [50](#)
- SMTP security [409](#)

TAG command [371](#)

tags

- [OCSPParms 459](#)

tasks

- configuring OSPF and RIP (IPv4 and IPv6)
 - steps for [205](#)
 - using TCP/IP with RACF
 - steps for [688](#)

TCBPOOLSIZE statement [620](#)

TCP [55](#), [389](#), [407](#)

TCP/IP for VM

- checksum testing [677](#)

TCPCOMP exec [682](#)

TCPIP configuration statements

- [ACBPOOLSIZE 527](#)
- [ADDRESSTRANSLOCATIONPOOLSIZE 528](#)

TCPIP configuration statements (*continued*)

- [ARPAGE 529](#)
- [ASSORTEDPARMS 529](#)
- [AUTOLOG 534](#)
- [BLOCK 535](#)
- [CBPOOLSIZE 536](#)
- [CTC DEVICE and LINK 539](#)
- [DATABUFFERLIMITS 537](#)
- [DATABUFFERPOOLSIZE 537](#)
- [ENVELOPEPOOLSIZE 564](#)
- [FILE 565](#)
- [FIXEDPAGESTORAGEPOOL 566](#)
- [FOREIGNIPCONLIMIT 567](#)
- [GATEWAY 568](#)
- [HOME 579](#)
- [ICMPERRORLIMIT 583](#)
- [INFORM 584](#)
- [INTERNALCLIENTPARMS 584](#)
- [IPROUTEPOOLSIZE 590](#)
- IUCV connections DEVICE and LINK
 - local [545](#)
 - remote [548](#)
- [KEEPALIVEOPTIONS 590](#)
- LAN Channel (LCS) DEVICE and LINK [551](#)
- [LARGEENVELOPEPOOLSIZE 591](#)
- [LESSTRACE 592](#)
- [MONITORRECORDS 593](#)
- [MORETRACE 595](#)
- [NCBPOOLSIZE 595](#)
- [NOSCREEN 596](#)
- [NOTRACE 597](#)
- [OBEY 597](#)
- [OSD DEVICE and LINK 553](#)
- [OSH DEVICE and LINK 559](#)
- [PACKETTRACESIZE 598](#), [600](#)
- [PENDINGCONNECTIONLIMIT 600](#)
- [PERMIT 601](#)
- [PORT 603](#), [606](#)
- [PRIMARYINTERFACE 606](#)
- [RCBPOOLSIZE 608](#)
- [RESTRICT 608](#)
- [ROUTERADV 609](#)
- [ROUTERADVPREFIX 611](#)
- [SCBPOOLSIZE 613](#)
- [SCREEN 614](#)
- [SKCBPOOLSIZE 614](#)
- [SMALLDATABUFFERPOOLSIZE 615](#)
- [SOMAXCONN 616](#)
- [START 618](#)
- [STOP 618](#)
- [SYSCONTACT 619](#)
- [SYSLOCATION 619](#)
- [TCBPOOLSIZE 620](#)
- [TIMESTAMP 621](#)
- [TINYDATABUFFERPOOLSIZE 621](#)
- [TN3270E 622](#)
- [TRACE 623](#)
- [TRACEONLY 625](#)
- [TRANSLATE 626](#)
- [UCBPOOLSIZE 627](#)
- [UDPQUEUELIMIT 627](#)
- [VIPA DEVICE and LINK 563](#)
- [VSWITCH CONTROLLER 628](#)

TCPIP DATA client configuration statements

TCPIP DATA client configuration statements (*continued*)

- ATSIGN [14](#)
- DOMAINLOOKUP [14](#)
- DOMAINORIGIN [15](#)
- DOMAINSEARCH [16](#)
- HOSTNAME [18](#)
- HOSTVERIFICATION [18](#)
- NSINTERADDR [19](#)
- NSPORTADDR [20](#)
- RESOLVERTIMEOUT [20](#)
- RESOLVERUDPRETRIES [20](#)
- RESOLVEVIA [21](#)
- SMTPSERVERID [21](#)
- TCPIPUSERID [22](#)
- TRACE RESLOVER [22](#)
- UFTSERVERID [23](#)
- USERDATA [23](#)
- VMFILETYPE [23](#)
- VMFILETYPEDEFAULT [24](#)
- TCPIP DATA file [13](#)
- TCPIP virtual machine
 - attaching network devices to [519](#)
- TCPIPUSERID statement [22](#)
- TCPLoad exec [681](#)
- TCPRUN exec [35](#)
- tcpTerminate [159](#)
- TCPTXT exec [681](#)
- Telnet server [584](#)
- TEMPERORETRIES statement [400](#)
- testing functions for TCP/IP [677](#)
- testing TCP/IP system configuration [25](#)
- TESTSITE program [31](#)
- threads
 - specifying number in configuration [148](#)
 - specifying with commThreads [136](#)
- timeLimit option [159](#)
- timeout
 - specifying [139](#)
- TIMESTAMP statement [621](#)
- TINYDATABUFFERPOOLSIZE statement [621](#)
- TN3270 [2](#)
- TN3270E
 - printer
 - configuring [353](#)
 - TAG command [371](#)
 - RSCS printer link [365](#)
 - statement [622](#)
 - TELNET support [2](#)
 - with InternalClientParms statement [587](#)
- TNSIMHPI TEXT file [589](#)
- TRACE
 - command [660](#)
 - statement
 - TCPIP server [623](#)
 - UFTD server [652](#)
- TRACE RESOLVER statement [22](#)
- TRACEONLY statement [625](#)
- trademarks [726](#)
- TRANSLATE statement
 - TCPIP server [626](#)
 - UFTD server [653](#)
- translation tables
 - Chinese DBCS [675](#)
 - converting to binary [675](#)

translation tables (*continued*)

- Hangeul DBCS [674](#)
- IBM [669](#)
- Japanese KanjiDBCS [674](#)
- search order [668](#)
- Two-way encryption formats [123](#)

U

- UCBPOOLSIZE statement [627](#)
- UDPQUEUELIMIT statement [627](#)
- UFTCMDS EXIT command [661](#)
- UFTCMDS EXIT statement [653](#)
- UFTD commands
 - IDENTIFY [657](#)
 - NSLOOKUP [658](#)
 - QUERY [658](#)
 - QUIT [659](#)
 - STOP [659](#)
 - TRACE [660](#)
 - UFTCMDSEXIT [661](#)
- UFTD CONFIG file [650](#)
- UFTD configuration statements
 - IDENTIFY [650](#)
 - MAXFILEBYTES [651](#)
 - NSLOOKUP [651](#)
 - PORT [652](#)
 - TRACE [652](#)
 - TRANSLATE [653](#)
 - UFTCMDS EXIT [653](#)
- updating
 - schema for native authentication [88](#)
- useAdvancedReplication option [160](#)
- useNativeAuth option [147](#), [161](#)
- user ID
 - defining for LDAP server [82](#)
- USERDATA statement [23](#)
- userNativeAuth option [89](#)
- userPassword attribute value
 - encrypting [172](#)
 - specifying encryption method for [150](#)
- using a user's own virtual machine [352](#)
- using translation tables [667](#)
- UTF-8 characters
 - mapping with Unicode [190](#)
- UTFD exit interfaces [655](#), [657](#), [661](#)
- UTFD virtual machine [35](#)
- UTFSERVERID statement [23](#)
- utilities
 - DB2PWDEN [172](#)
 - DS2LDIF [175](#)
 - LDAPXOP [181](#)

V

- validateincomingV2strings option [161](#)
- VERIFYCLIENT statement [404](#)
- VERIFYCLIENTDELAY statement [405](#)
- verifying
 - LDAP server [109](#)
- VIPA (virtual IP address)
 - interfaces [210](#)
- VIPA (Virtual IP Addressing)

- VIPA (Virtual IP Addressing) *(continued)*
 - configuring
 - backing up a TCP/IP stack [518](#)
 - TCPIP virtual machine [515](#)
- VIPA interfaces [210](#)
- Virtual Devices (VIPA) DEVICE and LINK statement [563](#)
- Virtual IP Addressing (VIPA)
 - configuring
 - backing up a TCP/IP stack [518](#)
 - TCPIP virtual machine [515](#)
- virtual machines
 - definitions [33](#)
 - optional [33](#)
 - required [33](#)
 - user's own [352](#)
- VIRTUAL_LINK statement [230](#), [232](#)
- VMFASM exec [679](#)
- VMFC exec [680](#)
- VMFHASM exec [679](#)
- VMFHLASM exec [679](#)
- VMFILETYPE statement
 - defining system parameters [23](#)
- VMFILETYPEDEFAULT statement [24](#)
- VMFPAS exec [679](#)
- VMNFS
 - server
 - configuration file statements [327](#), [332](#)
 - machine authorization [327](#)
 - using an ESM [327](#)
 - virtual machine [35](#)
- VMNFS command syntax [326](#)
- VMNFS CONFIG file
 - modifying [327](#)
 - syntax rules [328](#)
- VMSSL command syntax [470](#)
- VSWITCH CONTROLLER statement [628](#)

W

- waitingThreads option [136](#)
- WARNINGAGE statement [405](#)
- Web Browser FTP Support [78](#)
- Web servers
 - using native authentication with [99](#)
- welcome banner, FTP [69](#), [71](#)

X

- X.509 standard
 - digital certificate [118](#)



Product Number: 5741-A09

Printed in USA

SC24-6331-73

