OS/390 SecureWay Communications Server

IP Configuration

Version 2 Release 8

SC31-8513-03

OS/390 SecureWay Communications Server

IP Configuration

Version 2 Release 8

SC31-8513-03

Note:
Before using this information and the product it supports, be sure to read the general information under “Appendix J. Notices” on page 1203.

Fourth Edition (September 1999)

This edition applies to OS/390 V2R8 (Program Number 5647-A01).

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Department CGMD
P.O. Box 12195
Research Triangle Park, North Carolina 27709
U.S.A.

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):
1-800-227-5088

Internet e-mail:
usib2hpd@vnet.ibm.com

World Wide Web:
http://www.s390.ibm.com/os390

IBMLink:
CIBMORCF at RALVM17

IBM Mail Exchange:
USIB2HPD at IBMMAIL

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

Contents

Figures xxi

Tables xxiii

About This Book xxv
Who Should Use This Book xxv
How to Use This Book xxv
How This Book Is Organized xxv
Where to Find Related Information on the Internet xxv
How to Contact IBM Service xxvi

Summary of Changes xxvii

Part 1. Configuring the Base TCP/IP System 1

Chapter 1. Before You Begin 7
Overview for New Users 7
Configuration Data Sets and HFS Files 13
Overview of Data Sets and HFS Files 14
How CS for OS/390 Searches for Configuration Files 14
Information Specific to Data Sets in Configuration File Search Orders . . . 14
Search Order and Configuration Files for the TCP/IP Stack 16
Search Order and Configuration Files for TCP/IP Applications 19
MVS System Symbols 26
Changes to the Native TCP/IP Environment 27
Using Automatic Restart Manager 28
Customization Checklist 28
Security Considerations 31
IP Security (IPSEC) Considerations 31
Security Product Considerations 31
Migration Considerations 32

Chapter 2. Customization and Administration Overview 35
General Customization Procedure 35
Step 1: Install CS for OS/390 35
Verifying the Initial Installation 35
Step 2: Customize CS for OS/390 36
Step 3: Configuring VMCF and TNF 37
Step 4: Update the VTAM Application Definitions 40
Step 5: Test and Verify Your Configuration 40
Step 6: Accept the Product Installation 43
Cataloged Procedures and Configuration Data Sets 43
Updating Your Procedure Library 43
Cataloged Procedures 44
Configuration Data Sets 53
Customizing TCP/IP Messages 55
How to Access the Message Data Sets 55
Message Format 56
Rules for Customizing the Messages 56
Considerations for Multiple Instances of TCP/IP 57
Common INET Physical File System (C-INET PFS) 58
Port Management Overview 58
SMF Accounting Issues 64

Selecting a Stack 64
Specifying BPXPRMxx Values for a C-INET Configuration. 69

Chapter 3. Virtual IP Addressing 71
Introduction to VIPA. 71
Moving a VIPA (Upon Failure of TCP/IP) 72
Static vs. Dynamic VIPAs. 73
Using Static VIPAs 74
Configuring Static VIPAs for an OS/390 TCP/IP Stack 74
Configuring Static VIPAs for Enterprise Extender 76
Planning for Static VIPA Takeover and Takeback 77
Static VIPA and Routing Protocols 77
Using Dynamic VIPAs 77
Configuring the Dynamic VIPA Support. 78
Dynamic Movement of a VIPA (Dynamic VIPA Takeover Function). 78
Planning for Dynamic VIPA Takeover 79
Different Application Uses of IP Addresses and VIPAs 80
Overview of Dynamic VIPA Configuration and Operation 81
Planning for the Multiple Application Instance Scenario. 84
Planning for the Unique Application Instance Scenario 85
Choosing Which Form of Dynamic VIPA Support to Use 86
Resolution of Dynamic VIPA Conflicts 86
Dynamic VIPA Creation Results 86
Restart of the Original VIPADEFINE TCP/IP After a Failure 88
Movement of Unique Application Instance (BIND) 89
Movement of Unique APF-Authorized Application Instance (Utility-activated) . 89
Same Dynamic VIPA as VIPADEFINE and BIND()/UTILITY/IOCTL() 89
Other Considerations 90
Mixture of Types of Dynamic VIPAs within Subnets 90
MVS Failure and Sysplex Failure Management. 90
UDP and Dynamic VIPAs. 90
Example of Configuring Dynamic VIPAs 91
Verifying the Configuration 92
NETSTAT Support for Dynamic VIPAs 94
Dynamic VIPAs and Routing Protocols 95
OROUTED 95
OMPROUTE 96

Chapter 4. Configuring the TCP/IP Address Space 97
Update the TCP/IP Cataloged Procedure 97
TCP/IP Cataloged Procedure (TCPIPROC) 97
Using Output Data Sets 99
Specifying TCP/IP Address Space Parameters 99
Specify Configuration Statements in PROFILE.TCPIP 99
Changing Configuration Information 100
DEVICE and LINK Statements. 101
Relationship to VTAM Configuration 102
Routing Statements. 103
Multiple Network Attachments Support 104
Devices That Support ARP Offload 104
Summary of TCP/IP Configuration Statements 105
Sample Profile Configuration Data Set (SAMPPROF) 105
Cross-System Coupling Facility (XCF) Dynamics 114
PROFILE.TCPIP Configuration Statements 119
Summary of Values (Default, Minimum, and Maximum) for TCP/IP Configuration Statements 120

Statement Syntax 125
ARPAGE Statement 127
ASSORTEDPARMS Statement 128
ATMARPSV Statement 131
ATMLIS Statement 133
ATMPVC Statement 136
AUTOLOG Statement 137
BSDROUTINGPARMS Statement 140
DELETE Statement 144
DEVICE and LINK Statement—ATM Devices 148
DEVICE and LINK Statement—CLAW Devices 151
DEVICE and LINK Statement—CTC Devices 155
DEVICE and LINK Statement—HYPERchannel A220 Devices 158
DEVICE and LINK Statement—LAN Channel Station and OSA Devices . . 161
DEVICE and LINK Statement—MPCIPA Devices 169
DEVICE and LINK Statement—MPCOSA Devices 172
DEVICE and LINK Statement—MPCPTP Devices 175
DEVICE and LINK Statement—SNA LU0 Links 179
DEVICE and LINK Statement—SNA LU 6.2 Links 182
DEVICE and LINK Statement—X.25 NPSI Connections 185
DEVICE and LINK Statement—Virtual Devices 188
DEVICE and LINK Statement—3745/46 Channel DLC Devices 190
GATEWAY Statement 193
GLOBALCONFIG Statement 205
HOME Statement 207
INCLUDE Statement 211
IPCONFIG Statement 213
ITRACE Statement 220
KEEPALIVEOPTIONS Statement 222
PKTTRACE Statement 223
PORT Statement 229
PORTRANGE Statement 232
PRIMARYINTERFACE Statement 235
SACONFIG Statement 236
SMFCONFIG Statement 239
SMFPARMS Statement 241
SOMAXCONN Statement 243
START Statement 244
STOP Statement 246
TCPCONFIG Statement 247
TRANSLATE Statement 249
UDPCONFIG Statement 252
VIPADYNAMIC Statement 254

Chapter 5. Operator Commands and System Administration 259
Table of Commands 259
Administration Overview 261
Starting and Stopping TCP/IP Servers 261
START Command 261
STOP Command 261
Modifying Server Parameters 262
Modify Command 262
VARY Command—TCPIP Address Space 264
Security Considerations for the VARY Command 266
VARY TCPIP,,OBEYFILE 267
VARY TCPIP,,DROP 268

VARY TCPIP to START or STOP a Device 269
VARY TCPIP,,TELNET 270
VARY TCPIP,,PKTTRACE 271
DISPLAY Command—TCPIP Address Space 273
DISPLAY TCPIP,,HELP 274
DISPLAY TCPIP,,NETSTAT 277
DISPLAY TCPIP,,OMPROUTE 280
DISPLAY TCPIP,,SYSPLEX 281
DISPLAY TCPIP,,TELNET 283
Using the SMSG Interface 284

Chapter 6. Defining the TCP/IP Client System Parameters 285
Configuration Process 285
Summary of Statements in TCPIP.DATA 286
Sample TCPIP.DATA Data Set (TCPDATA) 287
TCPIP.DATA Configuration Statements 290
Syntax Conventions 291
ALWAYSWTO Statement 292
DATASETPREFIX Statement 293
DOMAINORIGIN Statement 294
HOSTNAME Statement 295
LOADDBCSTABLES Statement 296
MESSAGECASE Statement 298
NSINTERADDR Statement 299
NSPORTADDR Statement 300
RESOLVEVIA Statement 301
RESOLVERTIMEOUT Statement 302
RESOLVERUDPRETRIES Statement 303
SOCKDEBUG Statement 304
SOCKNOTESTSTOR Statement 305
TCPIPJOBNAME Statement 306
TRACE RESOLVER Statement 307
TRACE SOCKET Statement 308

Chapter 7. Configuring the Site Table 309
Configuration Process 310
Step 1: Update the HOSTS.LOCAL Data Set 310
Step 2: Run MAKESITE 311
MAKESITE Command 313
TESTSITE Command 316

Part 2. Configuring the Servers 317

Chapter 8. Configuring the SNALINK Environment 333
Understanding the SNALINK Environment 333
Configuration Process 334
Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP 334
Step 2: Update the SNALINK Cataloged Procedure 337
Step 3: Define the SNALINK Application to VTAM 338
Operating SNALINK 339
Stopping and Starting SNALINK 339
Sample Console 340
Determining the DLC Connection Status Using NETSTAT DEVLINKS . . . 341
Controlling the SNALINK LU0 Interface with the MODIFY Command 342
MODIFY Command—SNALINK LU0 343

Chapter 9. Configuring the SNALINK LU6.2 Interface 347
Configuration Process 347
Step 1: Specify DEVICE and LINK Statements in hlq.PROFILE.TCPIP . . . 347
Step 2: Update the SNALINK LU6.2 Cataloged Procedure 347
Step 3: Define the SNALINK LU6.2 Application to VTAM 348
Step 4: Update the SNALINK LU6.2 Configuration Data Set 349
SNALINK LU6.2 Configuration Statements 350
Statement Syntax 350
Statement Ordering 350
BUFFERS Statement 351
DEST Statement 352
LINK Statement 353
TRACE Statement 354
VTAM Statement 355
MODIFY Command—SNALINK LU 6.2 356

Chapter 10. Configuring the X.25 NCP Packet Switching Interface (NPSI) Server 361
Configuration Process 361
Step 1: Specify X.25 Configuration Statements in hlq.PROFILE.TCPIP . . . 362
Step 2: Update the X.25 NPSI Cataloged Procedure. 362
Step 3: Modify the X.25 NPSI Server Configuration Data Set 363
Step 4: Define the X.25 NPSI Configuration 365
Step 5: Define the X.25 NPSI Application to VTAM 368
Step 6: Define VTAM Switched Circuits 368
Configuring Cross-Domain Resources 370
X.25 NPSI Server Configuration Statements 370
Statement Syntax 370
ALTLINK Statement 371
BUFFERS Statement 373
DEST Statement 374
FAST Statement 375
LINK Statement 376
OPTIONS Statement 377
TIMERS Statement 379
TRACE Statement 380
VTAM Statement 382
MODIFY Command—X.25 NPSI Server 383

Chapter 11. Configuring the OS/390 UNIX Telnet Server 387
Installation Information 387
Starting, Stopping, and Administration of OS/390 UNIX Telnet 388
otelnetd 392
SMF Record Handling 394
BPX.DAEMON Considerations 394

Chapter 12. Configuring the Telnet Server 395
Configuration Options 395
Customizing the TCPIP Cataloged Procedure 396
Customizing Translate Tables for 3270 DBCS Transform Mode 397
Customizing the VTAM Configuration Data Set 397
Customizing Statements in the PROFILE Data Set 398
Profile Modifications. 398
General Rules for PROFILE Statements 400
Specifying the Stand-Alone PORT Statement 402
Specifying the TELNETPARMS Statements in the PROFILE Data set . . . 404

Specifying BEGINVTAM Statements in the PROFILE Data Set 429
Telnet PROFILE Example 457
Configuration Considerations 460
Choosing Telnet Solicitor or USSMSG Logon Panel 460
Considerations for Using TN3270 Enhanced Support (TN3270E) 480
Logmode Considerations 485
Telnet Device Name Parameters 486
Selection Sequence Considerations for VTAM Applications and LUs 486

Chapter 13. Managing the Telnet Server 489
Operating the Telnet Server Using the VARY TCPIP Command Set 489
VARY ACT Command 490
VARY INACT Command 491
VARY OBEYFILE Command 492
VARY QUIESCE Command 493
VARY RESUME Command 494
VARY STOP Command 495
Operating the Telnet Server Using the DISPLAY Command Set. 496
The Profile Group Displays 498
APPL Display Command 499
DEFAULTS Display Command 501
DEVICETYPE Display Command 503
HNGROUP Display Command 505
IPGROUP Display Command 507
LINKNAME Display Command 509
LUGROUP Display Command 510
LUMAP Display Command 512
PROFILE Display Command 513
WHEREUSED Display Command 515
The Connection Group Displays – CONNECTION Display Command 517
The Port Group Displays – WLM Display Command 520
The Server Group Displays - INACTLUS Display Command 521

Chapter 14. Configuring the File Transfer Protocol (FTP) Server 523
Configuration Process 523
Step 1: Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS Information . 523
Step 2: Update ETC.SERVICES 524
Step 3: Update the FTPD Cataloged Procedure 524
Step 4: Specify FTP Configuration Statements in FTP.DATA 527
Step 5: Configure the FTP Server for SMF (Optional) 533
Step 6: Configure the User-Written Exits (Optional) 536
Step 7: Specify Configuration Statements in TCPIP.DATA 539
Step 8: Install the SQL Query Function (Optional) and Access the DB2 Modules 539
Step 9: Update /etc/syslog.conf for the FTP Server 542
Implementing an Anonymous User 542
Identified User 542
Anonymous User 544
Security Considerations for the FTP Server 545
FTP.DATA Data Set Statements 546
ANONYMOUS Statement 547
ASATRANS Statement. 549
AUTOMOUNT Statement 550
AUTORECALL Statement 551
AUTOTAPEMOUNT Statement 552
BLKSIZE Statement. 553

BUFNO Statement 554
CCXLATE Statement 555
CHKPTINT Statement 556
CONDDISP Statement 557
CTRLCONN Statement 558
DATACLASS Statement 559
DB2 Statement 561
DB2PLAN Statement 562
DCBDSN Statement 563
DEST Statement 565
DIRECTORY Statement 566
DIRECTORYMODE Statement 567
FILETYPE Statement 568
INACTIVE Statement 569
JESLRECL Statement 570
JESOUTGETTO Statement 571
JESRECFM Statement 572
LRECL Statement 573
MGMTCLASS Statement 574
MIGRATEVOL Statement 575
PRIMARY Statement 576
QUOTESOVERRIDE Statement 577
RDW Statement 578
RECFM Statement 579
RETPD Statement 580
SBDATACONN Statement 581
SECONDARY Statement 582
SMF Statement 583
SMFAPPE Statement 585
SMFDEL Statement 586
SMFEXIT Statement 587
SMFJES Statement 588
SMFLOGN Statement 589
SMFREN Statement 590
SMFRETR Statement 591
SMFSQL Statement 592
SMFSTOR Statement 593
SPACETYPE Statement 594
SPREAD Statement 595
SQLCOL Statement 596
STARTDIRECTORY Statement 597
STORCLASS Statement 598
TRACE Statement 599
TRAILINGBLANKS Statement 600
UCSHOSTCS Statement 601
UCSSUB Statement 602
UCSTRUNC Statement 603
UMASK Statement 604
UNITNAME Statement 605
VOLUME Statement 606
WLMCLUSTERNAME Statement 607
WRAPRECORD Statement 608
XLATE Statement 609
Starting, Stopping, and Tracing the FTP Server 610
  Starting the FTP Server 610
  Stopping the FTP Server 611

Tracing the FTP Server 611
FTP Code Page Conversion 613
Code Page Conversions for the Control Connection 613
Code Page Conversions for the Data Connection 613

Chapter 15. Configuring the Trivial File Transfer Protocol Server 615
Considerations for OS/390 615

Chapter 16. Configuring the TIMED Daemon 619
Starting TIMED from OS/390 619
Starting TIMED as a Procedure 619

Chapter 17. Configuring the OS/390 Unix Remote Execution Server . . . 621
Installation Information 621
HFS Files for OS/390 UNIX REXECD 621
HFS Files for OS/390 UNIX RSHD 622
Setting up the OS/390 UNIX RSHD Installation Exit 622
OS/390 UNIX REXECD Command (orexecd) 623
OS/390 UNIX RSHD Command (orshd) 623

Chapter 18. Configuring the Remote Execution Server 625
Configuration Process 625
Step 1: Specify the AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP 625
Step 2: Determine Whether Remote Execution Client Will Send REXEC or RSH Commands 626
Step 3: Permit Remote Users to Access MVS Resources 626
Step 4: Update the Remote Execution Cataloged Procedure 627
Step 5: Create a User Exit Routine (Optional) 629
Modifying the Remote Execution Server Operating Parameters 632
MODIFY Command—Remote Execution Server 633

Chapter 19. Configuring the SMTP Server 635
Configuration Process 635
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP . . 635
Step 2: Update the SMTP Cataloged Procedure 636
Step 3: Customize the System CLIST and Modify PARMLIB Data Sets . . . 637
Step 4: Customize the SMTP Mail Headers (Optional) 638
Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional) 647
Step 6: Specify Configuration Statements in SMTP Configuration Data Set . 648
Step 7: Create an SMTP Security Table (Optional) 653
Step 8: Enable SMTP Domain Name Resolution 655
SMTP Configuration Data Set Statements 656
ALTNJEDOMAIN Statement 657
ALTTCPHOSTNAME Statement 658
BADSPOOLFILEID Statement 659
DBCS Statement 660
DEBUG Statement 663
FINISHOPEN Statement 664
GATEWAY Statement 665
INACTIVE Statement 667
IPMAILERADDRESS Statement 668
LISTENONADDRESS Statement 669
LOCALCLASS Statement 670
LOCALFORMAT Statement 671
LOG Statement 672

MAILER Statement 673
MAILFILEDSPREFIX Statement 675
MAILFILEUNIT Statement 676
MAILFILEVOLUME Statement 677
MAXMAILBYTES Statement 678
NJECLASS Statement 679
NJEDOMAIN Statement 680
NJEFORMAT Statement 681
NJENODENAME Statement 682
NOLOG Statement 683
OUTBOUNDOPENLIMIT Statement 684
PORT Statement 685
POSTMASTER Statement 686
RCPTRESPONSEDELAY Statement 687
RESOLVERRETRYINT Statement 688
RESTRICT Statement 689
RETRYAGE Statement 691
RETRYINT Statement 692
REWRITE822HEADER Statement 693
SECURE Statement 694
SMSGAUTHLIST Statement 695
SPOOLPOLLINTERVAL Statement 696
TEMPERRORRETRIES Statement 697
TIMEZONE Statement 698
WARNINGAGE Statement 699
Operating the SMTP Server 700
Using SMSG with SMTP 700
Forcing Re-resolution of Queued Mail 700

Chapter 20. Configuring OS/390 UNIX sendmail and Popper 701
Overview 701
Configuring OS/390 UNIX sendmail 703
The sendmail Samples Directory 703
Creating the Configuration File 704
Creating the Aliases File 706
Configuration Hints and Tips 707
Configuring Popper 708

Chapter 21. Configuring the Bind-Based Domain Name System (DNS) . . 709
DNS and BIND Overview 709
Domain Name Servers 710
Recommended Reading 712
Setting Up and Running the Name Server 713
Migrating to the Name Server 713
Configuring the Name Server 714
Configuring Host Resolvers: Name Server Considerations 718
Creating the Syslog File 718
Querying Name Servers 718
onslookup/nslookup Command 719
Configuring Host Resolvers: onslookup/nslookup Considerations 721
Diagnosing Problems 722
Checking Messages Sent to the Operators Console 722
Checking the Syslog Messages 723
Using Name Server Signals to Diagnose Problems 723
Using onslookup/nslookup to Diagnose Problems 723
Connection Optimization in a Sysplex Domain 723

Overview . . . 724
Configuring a Sysplex Domain for Connection Optimization . . . 731
Registering Your Own Applications . . . 736
Dynamic IP . . . 737
Overview . . . 737
Administering Dynamic Domains . . . 739
Migrating an Existing DNS Configuration to Dynamic IP . . . 740
RSA Encryption . . . 740
Configuring the DHCP Server for OS/390 . . . 742
Changing the DHCP Configuration File . . . 749
Configuring DHCP Server as DDNS Client Proxy . . . 772
Defining DHCP Proxy Authority . . . 776
Configuring the BINL Server . . . 778
Configuring a DDNS Server . . . 783
Control Entries, Resource Records, and Special Characters . . . 787
Control Entries . . . 787
Resource Records . . . 787
Special Characters . . . 791
Boot File Directives . . . 792
Sample Files . . . 794
Sample Forward Domain Data File . . . 794
Sample Reverse Domain Data File . . . 795
Sample Hints (Root Server) File . . . 795
Sample Loopback File . . . 797
Sample Boot File . . . 797
The named Daemon . . . 799
onslookup/nslookup—Querying A Name Server in Command Mode . . . 803
onslookup/nslookup—Issuing Multiple Queries in Interactive Mode . . . 805
nsupdate Command . . . 818
nsupdate Subcommands . . . 820
nsupdate Examples . . . 820
How an Administrator Removes and Locks Out a Host Name . . . 820
How an Administrator Creates an Alias for the Dynamic Zone . . . 821
Return Codes . . . 821

Chapter 22. Configuring Simple Network Management Protocol (SNMP) . . . 823
Processing SNMP Request . . . 823
Deciding on SNMP Security Needs . . . 824
Step 1: Configure the SNMP Agent (OSNMPD) . . . 825
Provide TCP/IP Profile Statements . . . 826
Provide Community-Based Security and Trap Destination Information . . . 827
Provide Community-Based and User-Based Security and Trap Destination Information . . . 830
Provide MIB Object Configuration Information . . . 850
Start the SNMP Agent (OSNMPD) . . . 852
Step 2: Configure the SNMP Commands . . . 855
Configure the NetView SNMP (SNMP) Command . . . 855
Configure the OS/390 UNIX SNMP (osnmp) Command . . . 861
Step 3: Configure the SNMP Subagents . . . 866
Step 4: Configure the ATM Open Systems Adapter 2 (ATM OSA-2) Support . . . 867
OSA/SF Prerequisites . . . 868
Required TCP/IP Profile Statements . . . 869
Multiple TCP/IP Instances Considerations . . . 869

Chapter 23. Configuring the Kerberos Server . . . 871
Configuration Process . . . 871

Step 1: Add PORT Statements to the hlq.PROFILE.TCPIP Data Set 871
Step 2: Update the Authentication Server Cataloged Procedure. 872
Step 3: Update the Remote DBA Server Cataloged Procedure 872
Step 4: Define the Kerberos Services in hlq.ETC.SERVICES 873
Step 5: Create and Update the Kerberos System Data Sets 873
Step 6: Authorize Kerberos Servers 875
Step 7: Build the Kerberos Database 875
Step 8: Store the Master Key 875
Step 9: Start the Kerberos Servers 875
Verifying the Kerberos Configuration. 877
Step 1: Set Up the Environment 878
Step 2: Register the Sample Service and the User 878
Step 3: Generate the Key Data Set for the Sample Service 879
Step 4: Transfer the Service Key Data Set to the Server 879
Step 5: Start the Sample Server 879
Step 6: Get the Initial Ticket. 880
Step 7: Run the Sample Client Program 880
Setting Up a Service or Client Application. 880
Setting Up a Service Application 880
Setting Up a Client Application. 881
Administrative Commands for the Kerberos Database 881
EXT@SRVT Command 883
KADMIN Command. 884
KDB@DEST Command 886
KDB@EDIT Command 887
KDB@INIT Command 889
KDB@UTIL Command. 890
KSTASH Command. 891

Chapter 24. Configuring the Remote Print Server (LPD) 893
Configuration Process 893
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP. . 893
Step 2: Update the LPD Server Cataloged Procedure 894
Step 3: Update the LPD Server Configuration Data Set. 896
Step 4: Create a Banner Page (Optional) 898
Statements for the LPD Server Configuration Data Set 899
Syntax Rules 899
DEBUG Statement 900
JOBPACING Statement 901
OBEY Statement. 902
SERVICE Statement 903
STEPLIMIT Statement. 913
UNIT Statement 914
VOLUME Statement 915
Operating the LPD Server Using the SMSG Interface 916

Chapter 25. Configuring the OROUTED Server. 917
Understanding OROUTED 917
Routing Information Protocol (RIP) 918
RIP Version 2 919
OROUTED Miscellaneous Features 920
RIP Input/Output Filters 920
RIP Routes. 921
OROUTED Gateways 921
Passive RIP Routes. 921
External RIP Routes 922

Active Gateways 922
OROUTED Gateways Summary 922
Configuration Process 923
Step 1: Configure the OROUTED Profile 923
Step 2: Update Configuration Statements in PROFILE.TCPIP 925
Step 3: Update the Resolver Configuration File. 926
Step 4: Update the OROUTED Cataloged Procedure (Optional) 926
Step 5: Specify the OROUTED Port Number in the SERVICES File 928
Step 6: Configure the Gateways File or Data Set (Optional) 928
Step 7: Configure and Start syslogd 934
Step 8: RACF-Authorize User IDs 934
OROUTED Parameters 934
Specifying Parameters 936
Starting OROUTED 937
Configuration Examples 937
Configuring a Passive Route 938
Configuring an External Route 939
Configuring an Active Gateway 939
Configuring a Point-to-Point Link 940
Configuring a Default Route 940
Configuring ORouteD with Enterprise Extender 940
Configuring OROUTED with VIPA 941
Configuring OROUTED to Split Traffic with VIPA 941
Configuring a Backup TCP/IP Stack with VIPA 943
Restoring a Primary OS/390 TCP/IP Stack with VIPA 944
Controlling OROUTED with the MODIFY Command 945

Chapter 26. Configuring OMPROUTE 949
Understanding OMPROUTE. 949
Open Shortest Path First (OSPF) 951
Routing Information Protocol (RIP) 952
The Configuration Process 953
Step 1: Create the OMPROUTE Configuration File 953
Step 2: Reserve the RIP UDP Port (If Using the RIP Protocol) 954
Step 3: Update the Resolver Configuration File. 954
Step 4: Update the OMPROUTE Cataloged Procedure (Optional) 955
Step 5: Specify the RIP UDP Port Number in the SERVICES File or Data Set (If Using the RIP Protocol) 955
Step 6: RACF-Authorize User IDs for Starting OMPROUTE 956
Step 7: Start syslogd 956
Step 8: Update the OMPROUTE Environment Variables (Optional) 956
Step 9: Create Static Routes (Optional) 956
Starting OMPROUTE 957
OMPROUTE Parameters 958
Controlling OMPROUTE with the MODIFY Command 958
Stopping OMPROUTE 959
Rereading the Configuration File 959
Enabling or Disabling the OMPROUTE Subagent 959
Changing the Cost of OSPF Links 960
Controlling OMPROUTE Tracing 960
OMPROUTE Configuration File 960
OSPF Configuration Statements 960
RIP Configuration Statements 976
Common Configuration Statements 987
Using OMPROUTE DISPLAY Commands. 993
All OSPF Configuration Information 993

Configured OSPF Areas and Ranges 995
Configured OSPF Interfaces 995
Configured OSPF Non-Broadcast, Multi-Access Networks 996
Configured OSPF Virtual Links 996
Configured OSPF Neighbors 997
OSPF Link State Advertisement 997
OSPF Area Statistics and Parameters 999
OSPF External Advertisements 1000
OSPF Area Link State Database 1000
OSPF Interface Statistics and Parameters 1001
OSPF Neighbor Statistics and Parameters 1003
OSPF Router Routes 1005
OSPF Link State Database Statistics 1006
OSPF Routing Protocol Statistics 1006
OMPROUTE Routing Table 1008
Route Expansion Information 1009
RIP Configuration Information 1010
Configured RIP Interfaces 1011
RIP Routes to Be Accepted 1011
RIP Interface Statistics and Parameters 1012
Global RIP Filters 1013
Configuring Interfaces to OMPROUTE 1014
Configuring Point-to-Point Interfaces 1015
Configuring MPC, XCF and IUT SameHost Interfaces 1015
Configuring Non-Broadcast Network Interfaces 1016
Configuring Broadcast Network Interfaces 1017
Configuring VIPA Interfaces 1017

Chapter 27. Using OSPF 1021
OSPF Routing Summary 1021
Designated Router 1022
Configuring OSPF 1022
Setting OSPF Router IDs 1023
Defining Backbone and Attached OSPF Areas 1023
Setting OSPF Interfaces 1026
Setting Non-Broadcast Network Interface Parameters 1028
Configuring Wide Area Subnetworks 1028
Enabling AS Boundary Routing 1029
Configuring OSPF over ATM 1030
Configuring OSPF over ATM Native 1030
Other Configuration Tasks 1030
Converting from RIP to OSPF 1033

Chapter 28. Configuring the NCPROUTE Server 1035
Understanding NCPROUTE 1035
Environment 1036
NCPROUTE Operation 1037
NCPROUTE Gateways Summary 1039
RIP Input/Output Filters 1039
Configuration Process 1039
Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP 1040
Step 2: Configure VTAM and SNALINK Applications 1041
Step 3: Configure the IP over CDLC DEVICE and LINK Statements 1042
Step 4: Update the NCPROUTE Cataloged Procedure 1042
Step 5: Update hlq.ETC.SERVICES 1045
Step 6: Configure the Host-Dependent NCP Clients 1045

Step 7: Configure the NCPROUTE Profile Data Set 1049
Step 8: Configure the NCPROUTE Gateways Data Set (Optional) 1051
Step 9: Define a Directly Connected Host Route for the NCST Session . . 1061
Controlling the NCPROUTE Address Space with the MODIFY Command . 1061
MODIFY Command—NCPROUTE Address Space 1062

Chapter 29. Configuring the OS/390 Unix PORTMAP Address Space . . 1065
Configuration Process 1065
Step 1: Specify the PORT Statements in PROFILE.TCPIP 1065
Step 2: Update the PORTMAP Cataloged Procedure 1065
Starting the PORTMAP Address Space 1066

Chapter 30. Configuring the PORTMAP Address Space 1067
Configuration Process 1067
Step 1: Specify the AUTOLOG and PORT Statements in PROFILE.TCPIP 1067
Step 2: Update the PORTMAP Cataloged Procedure 1068
Step 3: Define the Data Set for Well-Known Procedure Names 1068
Starting the PORTMAP Address Space 1071

Chapter 31. Configuring the NCS Interface 1073
Understanding the NCS Interface 1073
Understanding the LLBD Server 1073
Understanding the NRGLBD Server 1073
Configuration Process 1074
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP. 1074
Step 2: Update the NRGLBD Cataloged Procedure 1074
Step 3: Update the LLBD Cataloged Procedure 1075

Chapter 32. Configuring the Network Database (NDB) System 1077
Configuration Process 1077
Step 1: Update the NDB Setup Sample Job 1077
Step 2: Run the NDB Setup Job 1079
Step 3: Update and Install the DB2 Sample Connection Exit Routine. . . 1080
Step 4: Update the PORTS Cataloged Procedure 1081
Step 5: Update the PORTC Cataloged Procedure 1082
Step 6: Create the NDB Clients 1084
Starting NDB 1089

Chapter 33. Configuring the Miscellaneous (MISC) Server 1091
Understanding the MISC Server 1091
Discard Protocol 1091
Echo Protocol 1091
Character Generator Protocol 1091
Configuration Process 1092
Step 1: Specify AUTOLOG and PORT Statements in PROFILE.TCPIP . . 1092
Step 2: Update the MISC Server Cataloged Procedure (MISCSERV) . . 1093

Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent 1095
Configuring the Service Policy Agent 1095
Overview 1095
The Policy Profile Control Block (ProfileData) 1096
The Policy Configuration File 1096
Starting the Service Policy Agent 1109
Starting Pagent from the OS/390 Shell 1109
Starting Pagent as a Started Task 1110

Stopping the Service Policy Agent 1111
Configuring the SLA Subagent 1111
SLA Subagent Performance Monitoring 1111
Starting the SLA Subagent 1117
Starting the SLA Subagent from the OS/390 Shell 1117
Starting the SLA Subagent as a Started Task 1118
Stopping the SLA Subagent 1119
Controlling the SLA Subagent with the Modify Command 1119
Security Information 1120

Chapter 35. Configuring the RSVP Agent 1121
Description of RSVP 1121
Configuring the RSVP Agent 1122
The RSVP Configuration File 1122
LogLevel Statement 1123
TcpImage Statement 1124
Interface Statement 1125
RSVP Statement 1127
Starting the RSVP Agent 1129
Starting RSVP from the OS/390 Shell 1129
Starting RSVP as a Started Task 1129
Stopping RSVP 1130
Security Information 1130

Part 3. Using Translation Tables 1131

Chapter 36. Using Translation Tables 1133
SBCS Translation Table Hierarchy 1133
Customizing SBCS Translation Tables 1135
Syntax Rules for SBCS Translation Tables 1136
SBCS Country Translation Tables 1136
ISO-8 and IBM PC Interpretations for ASCII and EBCDIC Code Points . . 1137
DBCS Translation Table Hierarchy 1138
Usage Notes for the TRANSLATE Option for the FTP Client 1140
Telnet 3270 DBCS Transform Mode Codefiles 1140
Customizing DBCS Translation Tables 1141
DBCS Country Translation Tables 1141
Converting Translation Tables to Binary 1144

Part 4. Appendixes 1149

Appendix A. HFS File Requirements 1151
OS/390 UNIX Hierarchical File System Concepts 1151
The Root File System 1151
Mounting the CS for OS/390 HFS 1151
Location of CS for OS/390 Files in the HFS 1152
Directories or Files 1153

Appendix B. SMF Records 1163
Standard Subtype Record Numbers 1163
Telnet Server SMF Record Layout 1164
FTP Server SMF Record Layout 1164
SMF Record Layout for API Calls 1165
SMF Record Layout for FTP Client Calls 1166
SMF Record Layout for Telnet Client Calls 1167

SMF Record Layout for TCPIPSTATISTICS 1168

Appendix C. Creating a Keyring File for the Telnet Server 1173
Using the GSKKYMAN Utility 1173
Where to Find GSKKYMAN 1174
Using GSKKYMAN with the Telnet Server 1174
Self-signed Server Certificates 1175
Self-signed Client Certificates 1175

Appendix D. Related Protocol Specifications (RFCs) 1177

Appendix E. Syslog Daemon 1183
Files 1184
Starting syslogd 1184
Operation 1185
Configuration Statements 1185
Facility Names 1187
Priority Codes 1187
Destinations 1188
Usage Notes 1188
Application Considerations 1189

Appendix F. Setting up the inetd Configuration File 1191

Appendix G. Protocol Number and Port Assignments 1193
Protocol Assignments from /etc/protocol 1193
Port Assignments 1193
PROFILE.TCPIP Port Assignments 1193
/etc/services Port Assignments 1194

Appendix H. How to Read a Syntax Diagram 1197
Symbols and Punctuation 1197
Parameters 1197
Syntax Examples 1197
Longer Than One Line 1198
Required Operands 1198
Choose One Required Item from a Stack 1198
Optional Values 1198
Choose One Optional Operand from a Stack 1198
Repeating an Operand 1199
Selecting More Than One Operand 1199
Nonalphanumeric Characters 1199
Blank Spaces in Syntax Diagrams 1199
Default Operands 1199
Variables 1200
Syntax Fragments 1200

Appendix I. Information Apars 1201
IP Information Apars 1201

Appendix J. Notices 1203
Trademarks 1206

Bibliography 1209
SecureWay Communications Server for OS/390 Publications 1209
Related Publications 1209

Softcopy Information 1209
Planning 1209
Resource Definition, Configuration, and Tuning 1210
Operation 1210
Customization 1210
Writing Application Programs 1210
Diagnosis 1211
Messages and Codes 1212
APPC Application Suite 1212
Multiprotocol Transport Networking (MPTN) Architecture Publications . . . 1212
Redbooks 1212

Index 1215

Readers' Comments — We'd Like to Hear from You 1231

Figures

1. Resolver Components and Related Configuration Files in OS/390 UNIX and TCP/IP Environments 20
2. Generic Server. 59
3. Server with Affinity for a Specific Transport Provider 60
4. Example of Binding an Application to a Specific Transport Provider . . . 60
5. REXX Program to Switch TSO User to Another TCP/IP Stack 67
6. Selecting Configuration Data Sets. 68
7. Sharing DATASETPREFIX 69
8. SYS1.PARMLIB(BPXPRMxx) for Converged Sockets 70
9. Single VIPA Configuration. 76
10. Example of Network Connectivity Using Variable Subnetting 199
11. Example of Equal-cost Multipath Routes 201
12. Source VIPA Usage Chart 210
13. SNALINK Environment Interfaces 334
14. SNA DLC Link 335
15. APPL Statement for SNALINK 339
16. SNALINK Console Example 340
17. APPL Statement for SNALINK LU6.2 348
18. OS/390 UNIX Terminal Attachment Paths 389
19. Example TCPIP PROFILE Segment 458
20. USS Message Layout in Storage 470
21. Generic and Specific Device Pool Example 484
22. FTPD Server Before User Enters Login Information 542
23. FTPD Server After User Enters Login Information 543
24. FTP User Attempt to Violate Access Authorization 543
25. Example of a TCP-to-NJE Mail Gateway 665
26. Sender MUA Transmits the Message to sendmail 701
27. sendmail Transmits the Message to an Intermediate SMTP Server . . . 702
28. A sendmail Daemon Receives the Message from an SMTP Client 702
29. Sendmail Delivers the Message to the Local Recipient 702
30. Receiver's MUA has Direct Access to the Mail Spool File 703
31. Receiver's MUA Retrieves the Message over a POP3 Connection with a Popper Daemon 703
32. Name Resolution to a Sysplex 725
33. Address Association with mvsplex.mycorp.com 727
34. Address Association with myserver 728
35. Sample Network 812
36. Overview of SNMP Support 823
37. SNMPD.CONF File - Part 1 of 3 843
38. SNMPD.CONF File - Part 2 of 3 844
39. SNMPD.CONF File - Part 3 of 3 845
40. Subagent Connection to OSA/SF 870
41. Sample OROUTED Configuration File 925
42. Sample Portion of Services File 928
43. Example Commands to Start Multiple Copies of OROUTED 937
44. OROUTED Configuration Example 938
45. Configuring an Active Gateway. 939
46. Single VIPA Configuration. 942
47. Multiple VIPA Configuration 943
48. Portion of the Services File Relevant to OMPROUTE 956
49. OSPF Areas. 1024
50. OSPF Routing Hierarchy 1032
51. NCPROUTE Environment. 1036

52. NCPROUTE Example Configuration 1040
53. NCPROUTE Data Sets Relationship 1051
54. NCPROUTE Configuration Example 1058
55. Configuring an Active Gateway 1059
56. Syslogd Operation 1183
57. Sample /etc/syslog.conf File 1186
58. Adding Applications to /etc/inetd.conf 1191
59. Setting Traces in /etc/inetd.conf 1191

Tables

1. Overview of Functions Available in SecureWay Communications Server for OS/390. 7
2. Resolver Configuration Data Sets and Files 21
3. Checklist of Customization Tasks 29
4. TCP/IP Cataloged Procedures 44
5. TCP/IP Configuration Data Sets 53
6. How Your Own Socket Programs Select a Stack 66
7. Summary of Dynamic VIPA Creation Results. 86
8. Summary of Min, Max, and Default Values for TCP/IP Configuration Statements 120
9. Summary of TCP/IP Address Space Configuration Statements 124
10. BSDROUTINGPARMS Modification Methods. 142
11. Table of Commands 259
12. Servers or Address Spaces That Support the MVS Modify Command . . 262
13. Summary of TCPIP.DATA Configuration Statements 286
14. Summary of SNALINK LU6.2 Configuration Statements. 349
15. Summary of X.25 NPSI Server Configuration Statements 363
16. General Rules for PROFILE Statements 400
17. Variables Substituted for USSMSG 470
18. Variables Substituted for USSMSG 473
19. Explanation of Generic and Specific Device Pools. 484
20. Device Type and Logmode Table 486
21. Summary of FTP Server Configuration Statements 528
22. Summary of FTP Server SMF Statements. 534
23. Summary of FTP Server TCPIP.DATA Statements 539
24. Summary of SMTP Configuration Statements 648
25. Required and Recommended m4 Items 705
26. Sendmail Permission Table 707
27. Settings That Affect onslookup/nslookup Operation 720
28. DHCP Server Configuration 738
29. Data Formats for DHCP Options 761
30. Base Options 761
31. Options Affecting the Operation of the IP Layer on a Per Host Basis . . 763
32. IP Layer Parameters Per Interface Options 764
33. Link Layer Parameters Per Interface Options 764
34. TCP Parameter Options 765
35. Application and Service Parameter Options 765
36. DHCP Extensions Options 767
37. DHCP Load Balancing Options. 770
38. IBM-Specific Options 771
39. Security Advantages and Disadvantages 825
40. Summary of Kerberos Database Commands. 881
41. Summary of LPD Server Configuration Statements 896
42. OROUTED Gateways Summary 922
43. ORouteD Parameters 936
44. Sample Costs for OSPF Links 1027
45. NCPROUTE Gateways Summary. 1039
46. NCPRoute Options 1057
47. NCPROUTE Modify Parameters 1063
48. Monitor Control and Monitor Status Object Bit Values 1114
49. SBCS Translation Table Hierarchy 1133
50. Translation Table Members for Telnet Client and Non-Telnet SBCS Applications 1136

51. SBCS Translation Table Members for Telnet 3270 DBCS Transform Support 1137
52. ISO-8 Interpretations for Certain ASCII and EBCDIC Code Points . . . 1137
53. IBM PC Interpretations for Certain ASCII and EBCDIC Code Points . . 1138
54. DBCS Translation Table Hierarchy 1138
55. Translation Table Members for DBCS Applications. 1141
56. CS for OS/390 Parts 1152
57. Standard Subtype Record Numbers 1163
58. Telnet Server SMF Record Format 1164
59. FTP Server SMF Record Format 1164
60. API Call SMF Record Format 1165
61. FTP Client SMF Record Format 1166
62. Telnet Client SMF Record Format. 1167
63. SMF Record Layout for TCPIPSTATISTICS 1168
64. IP Information Apars. 1201

About This Book

This book describes how to configure the address spaces, servers, and applications available in SecureWay Communications Server for OS/390. This book also describes how to customize and administer CS for OS/390 for your specific needs.

For comments and suggestions about this book, use the comment form located at the back of this book. This form gives instructions for submitting your comments by mail, by fax, or electronically.

CS for OS/390 is an integral part of the OS/390® V2R8 family of products. For an overview and mapping of the documentation available for OS/390 V2R8, refer to the *OS/390 Information Roadmap.*

Who Should Use This Book

This book is intended for programmers and system administrators who are familiar with TCP/IP, MVS™, OS/390 UNIX®, and the Time Sharing Option Extensions (TSO/E).

How to Use This Book

Use this book to perform the following tasks:

- Configure CS for OS/390
- Customize and administer CS for OS/390

How This Book Is Organized

Part 1, Configuring the Base TCP/IP System, contains general information about configuring the base TCP/IP system. “Chapter 5. Operator Commands and System Administration” on page 259 contains information about and pointers to some useful commands included in this book.

Part 2, Configuring the Servers, contains chapters that explain the server configuration process for CS for OS/390:

Part 3, Using Translation Tables, describes how to use the translation tables.

Part 4, Appendixes, provides additional information.

Where to Find Related Information on the Internet

You may find the following information helpful.

You can read more about VTAM, TCP/IP, OS/390, and IBM on these Web pages. For up-to-date information about Web addresses, please refer to informational APAR II11334.

Home Page Web address
IBM SecureWay Communications Server product
http://www.software.ibm.com/network/commserver/

IBM SecureWay Communications Server support
http://www.software.ibm.com/network/commserver/support/
OS/390 http://www.ibm.com/os390/
OS/390 Internet Library
http://www.ibm.com/s390/os390/bkserv/
IBM Systems Center publications
http://www.ibm.com/redbooks
IBM Systems Center flashes
http://www.ibm.com/support/techdocs
VTAM and TCP/IP
http://www.software.ibm.com/network/commserver/about/csos390.html
IBM http://www.ibm.com

For definitions of the terms and abbreviations used in this book, you can view or download the latest *IBM Networking Softcopy Glossary* at the following Web address:

http://www.networking.ibm.com/nsg/nsgmain.htm

Note: Any pointers in this publication to websites are provided for convenience only and do not in any manner serve as an endorsement of these websites.

How to Contact IBM Service

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM® Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

Summary of Changes

Summary of Changes
for SC31-8513-03
SecureWay® Communications Server for OS/390 V2R8

This edition contains new and changed information that is indicated by vertical lines in the left margin.

New Information

- New CFGPRINT DD statement in the TCP/IP sample started procedure, TCPIPROC.
- New sections explaining modification steps have been added to the TCP/IP Profile configuration statements.
- In "Chapter 5. Operator Commands and System Administration" on page 259, command cross-references were added.
- "Appendix G. Protocol Number and Port Assignments" on page 1193 was added to Part 4.
- TN3270 DNS Client name to display.
- Telnet enhancements (TKOSPECLU statement).
- Service policy enhancements, including Service Level Agreement Subagent.
- New chapter about configuring the RSVP Agent.
- New Dynamic VIPA TCP/IP profile statements: VIPADYNAMIC, VIPABACKUP, VIPADEFINE, VIPARANGE, ENDVIPADYNAMIC.
- "Chapter 3. Virtual IP Addressing" on page 71, which includes VIPA takeover information.
- For enhanced multipath load balancing, new parameters and subparameters have been added to the IPCONFIG Profile statement. The new subparameters PERCONNECTION and PERPACKET have been added to the MULTIPATH parameter. The new subparameters, NOFWDMULTIPATH and FWDMULTIPATH PERPACKET have been added to the DATAGRAMFWD parameter.
- TN3270–SSL client authentication updates.
- D TCPIP,,NETSTAT,VIPADyn report option to display status of Dynamic VIPAs for a stack.
- D TCPIP,,SYSPLEX,VIPADyn report option to display status of all Dynamic VIPAs in a sysplex.
- Added MPCOSA device and OSAFDDI/OSAENET links.
- Automatic Restart Manager support.
- New GLOBALCONFIG TCP/IP Profile statement.

Changed Information

- The term eNetwork™ is replaced by SecureWay as part of our product name. The new name is SecureWay Communications Server.
- The Bibliography has been revised to show book number dash levels and delivery format.

Moved Information

- In "Chapter 4. Configuring the TCP/IP Address Space" on page 97, the Summary of TCP/IP Configuration Statements table has been moved to PROFILE.TCPIP Configuration Statements and renamed.

Summary of Changes
for SC31-8513-02
eNetwork Communications Server for OS/390 V2R7

The following new information has been added:

- MPCIPA DEVICE statement and IPAQGNET LINK statement for the OSA-Express feature.
- PATHMTUDISCOVERY, NOPATHMTUDISCOVERY, DYNAMICXCF, and NODYNAMICXCF parameters on the IPCONFIG statement.
- CACHINFO parameter on the D TCPIP,,NETSTAT command.
- HNGROUP/ENDHNGROUP telnet statements and new parameters on D TCPIP,,TELNET command.
- DHCP ServerType and ImageServer statements, and BINL server information.
- ROUTESA_CONFIG, Send_Only, and filter statements for OMPROUTE.
- MODIFY ROUTESA command to enable or disable the OMPROUTE subagent.
- SNMPv3 information to replace the SNMPv2u information.
- The automatic support of MVS system symbols in the PROFILE.TCPIP and OBEYFILE data sets, and the EZACFSM1 utility for translating MVS system symbols in other configuration files such as TCPIP.DATA.
- Service policy agent information

Summary of Changes
for SC31-8513-01
eNetwork Communications Server for OS/390 V2R6

The following new chapters have been added:

- Configuring Sendmail
- Configuring OMPROUTE
- Using OSPF

The chapter for the DB2-based DNS server was removed. Changed information includes:

- TELNETPARMS and BEGINVTAM profile statements
- D TCPIP,,TELNET
- GATEWAY statement
- TCPCONFIG statement

Note: As part of the name change of OpenEdition® to OS/390 UNIX System Services, occurrences of OS/390 OpenEdition have been changed to OS/390 UNIX System Services or its abbreviated name, OS/390 UNIX. OpenEdition may continue to appear in messages, panel text, and other code with OS/390 UNIX System Services.

Summary of Changes
for SC31-8513-00
eNetwork Communications Server for OS/390 V2R5

This was the first edition of this book. It contained information previously presented in the:

- *TCP/IP for MVS: Customization and Administration Guide* (SC31-7134-04), which supports TCP/IP Version 3 Release 2 for MVS

- *OS/390 TCP/IP OpenEdition Configuration Guide* (SC31-8304-00), which provided OpenEdition function for TCP/IP in the OS/390 environment.

Part 1. Configuring the Base TCP/IP System

Chapter 1. Before You Begin 7
Overview for New Users 7
Configuration Data Sets and HFS Files 13
Overview of Data Sets and HFS Files 14
How CS for OS/390 Searches for Configuration Files 14
Information Specific to Data Sets in Configuration File Search Orders . . . 14
Explicit Data Set Allocation 14
Dynamic Data Set Allocation 15
Search Order and Configuration Files for the TCP/IP Stack 16
PROFILE.TCPIP Search Order 17
TCPIP.DATA Search Order 18
Search Order and Configuration Files for TCP/IP Applications 19
Resolver Configuration Files 19
TCPIP.DATA 23
STANDARD.TCPXLBIN 23
HOSTS.SITEINFO 23
HOSTS.ADDRINFO 24
ETC.PROTO 24
ETC.SERVICES 24
Examples 25
MVS System Symbols 26
Changes to the Native TCP/IP Environment 27
Using Automatic Restart Manager 28
Customization Checklist 28
Security Considerations 31
IP Security (IPSEC) Considerations 31
Security Product Considerations 31
Migration Considerations 32

Chapter 2. Customization and Administration Overview 35
General Customization Procedure 35
Step 1: Install CS for OS/390 35
Verifying the Initial Installation 35
Step 2: Customize CS for OS/390 36
Making SYS1.PARMLIB Changes 36
Step 3: Configuring VMCF and TNF 37
Restartable Subsystems 37
Non-Restartable Subsystems 38
VMCF Commands 38
Common VMCF Problems 39
IUCV/VMCF Considerations 39
Step 4: Update the VTAM Application Definitions 40
Step 5: Test and Verify Your Configuration 40
Start the TCPIP Address Space 41
Verify Host Name and Address Configuration 41
Verify the X Window System Installation 42
Step 6: Accept the Product Installation 43
Cataloged Procedures and Configuration Data Sets 43
Updating Your Procedure Library 43
Naming Your Procedure Members 44
Specifying Job Step Wait Time 44
Cataloged Procedures 44
Configuration Data Sets 53

Customizing TCP/IP Messages 55
How to Access the Message Data Sets 55
Message Format 56
Message Text 56
Message Format 56
Rules for Customizing the Messages 56
Considerations for Multiple Instances of TCP/IP 57
Common INET Physical File System (C-INET PFS) 58
Port Management Overview 58
Generic Server Versus Server with Affinity for a Specific Transport Provider 59
Port Number Conflicts in a C-INET Environment 60
Port Reservation across Multiple Transport Providers 63
SMF Accounting Issues 64
Selecting a Stack 64
Standard Servers and Clients 65
Non-Standard Servers and Clients 65
TCP/IP TSO Clients 66
Selecting Configuration Data Sets 67
Sharing Resolver Configuration Data Sets 68
Specifying BPXPRMxx Values for a C-INET Configuration 69

Chapter 3. Virtual IP Addressing 71
Introduction to VIPA 71
Moving a VIPA (Upon Failure of TCP/IP) 72
Static vs. Dynamic VIPAs 73
Using Static VIPAs 74
Configuring Static VIPAs for an OS/390 TCP/IP Stack 74
Configuring Static VIPAs for Enterprise Extender 76
Planning for Static VIPA Takeover and Takeback 77
Static VIPA and Routing Protocols 77
Using Dynamic VIPAs 77
Configuring the Dynamic VIPA Support 78
Dynamic Movement of a VIPA (Dynamic VIPA Takeover Function) 78
Planning for Dynamic VIPA Takeover 79
Different Application Uses of IP Addresses and VIPAs 80
Overview of Dynamic VIPA Configuration and Operation 81
Configuration of Dynamic VIPAs 81
Multiple Application Instance Scenario 81
Unique Application Instance Scenario 82
Using the 'SIOCSVIPA' IOCTL Command 82
Using the EZBXFDVP Utility 83
Planning for the Multiple Application Instance Scenario 84
Planning for the Unique Application Instance Scenario 85
Choosing Which Form of Dynamic VIPA Support to Use 86
Resolution of Dynamic VIPA Conflicts 86
Dynamic VIPA Creation Results 86
Restart of the Original VIPADEFINE TCP/IP After a Failure 88
Movement of Unique Application Instance (BIND) 89
Movement of Unique APF-Authorized Application Instance (Utility-activated) . 89
Same Dynamic VIPA as VIPADEFINE and BIND()/UTILITY/IOCTL() 89
Other Considerations 90
Mixture of Types of Dynamic VIPAs within Subnets 90
MVS Failure and Sysplex Failure Management 90
UDP and Dynamic VIPAs 90
Example of Configuring Dynamic VIPAs 91

Verifying the Configuration 92
NETSTAT Support for Dynamic VIPAs 94
Dynamic VIPAs and Routing Protocols 95
OROUTED 95
OMPROUTE 96

Chapter 4. Configuring the TCP/IP Address Space 97
Update the TCP/IP Cataloged Procedure 97
TCP/IP Cataloged Procedure (TCPIPROC) 97
Using Output Data Sets 99
Specifying TCP/IP Address Space Parameters 99
Specify Configuration Statements in PROFILE.TCPIP 99
Changing Configuration Information 100
DEVICE and LINK Statements 101
Relationship to VTAM Configuration 102
Routing Statements 103
OMPROUTE 103
OROUTED 103
Multiple Network Attachments Support 104
Devices That Support ARP Offload 104
Summary of TCP/IP Configuration Statements 105
Sample Profile Configuration Data Set (SAMPPROF) 105
Cross-System Coupling Facility (XCF) Dynamics 114
Examples of Definitions Generated by XCF Dynamics 116
PROFILE.TCPIP Configuration Statements 119
Summary of Values (Default, Minimum, and Maximum) for TCP/IP Configuration Statements 120
Statement Syntax 125
ARPAGE Statement 127
ASSORTEDPARMS Statement 128
ATMARPSV Statement 131
ATMLIS Statement 133
ATMPVC Statement 136
AUTOLOG Statement 137
BSDROUTINGPARMS Statement 140
DELETE Statement 144
DEVICE and LINK Statement—ATM Devices 148
DEVICE and LINK Statement—CLAW Devices 151
DEVICE and LINK Statement—CTC Devices 155
DEVICE and LINK Statement—HYPERchannel A220 Devices 158
DEVICE and LINK Statement—LAN Channel Station and OSA Devices . . 161
DEVICE and LINK Statement—MPCIPA Devices 169
DEVICE and LINK Statement—MPCOSA Devices 172
DEVICE and LINK Statement—MPCPTP Devices 175
DEVICE and LINK Statement—SNA LU0 Links 179
DEVICE and LINK Statement—SNA LU 6.2 Links 182
DEVICE and LINK Statement—X.25 NPSI Connections 185
DEVICE and LINK Statement—Virtual Devices 188
DEVICE and LINK Statement—3745/46 Channel DLC Devices 190
GATEWAY Statement 193
GLOBALCONFIG Statement 205
HOME Statement 207
INCLUDE Statement 211
IPCONFIG Statement 213
ITRACE Statement 220
KEEPALIVEOPTIONS Statement 222

PKTTRACE Statement 223
PORT Statement 229
PORTRANGE Statement 232
PRIMARYINTERFACE Statement 235
SACONFIG Statement 236
SMFCONFIG Statement 239
SMFPARMS Statement 241
SOMAXCONN Statement 243
START Statement 244
STOP Statement 246
TCPCONFIG Statement 247
TRANSLATE Statement 249
UDPCONFIG Statement 252
VIPADYNAMIC Statement 254

Chapter 5. Operator Commands and System Administration 259
Table of Commands 259
Administration Overview 261
Starting and Stopping TCP/IP Servers 261
START Command 261
STOP Command 261
For each server with outstanding calls to TCP/IP 261
For each connected server that does not have outstanding calls 262
Modifying Server Parameters 262
Modify Command 262
VARY Command—TCPIP Address Space 264
Security Considerations for the VARY Command 266
VARY TCPIP,,OBEYFILE 267
VARY TCPIP,,DROP 268
VARY TCPIP to START or STOP a Device 269
VARY TCPIP,,TELNET 270
VARY TCPIP,,PKTTRACE 271
DISPLAY Command—TCPIP Address Space 273
DISPLAY TCPIP,,HELP 274
DISPLAY TCPIP,,NETSTAT 277
DISPLAY TCPIP,,OMPROUTE 280
DISPLAY TCPIP,,SYSPLEX 281
DISPLAY TCPIP,,TELNET 283
Using the SMSG Interface 284

Chapter 6. Defining the TCP/IP Client System Parameters 285
Configuration Process 285
Summary of Statements in TCPIP.DATA 286
Sample TCPIP.DATA Data Set (TCPDATA) 287
TCPIP.DATA Configuration Statements 290
Syntax Conventions 291
ALWAYSWTO Statement 292
DATASETPREFIX Statement 293
DOMAINORIGIN Statement 294
HOSTNAME Statement 295
LOADDBCSTABLES Statement 296
MESSAGECASE Statement 298
NSINTERADDR Statement 299
NSPORTADDR Statement 300
RESOLVEVIA Statement 301
RESOLVERTIMEOUT Statement 302

RESOLVERUDPRETRIES Statement 303
SOCKDEBUG Statement 304
SOCKNOTESTSTOR Statement 305
TCPIPJOBNAME Statement 306
TRACE RESOLVER Statement 307
TRACE SOCKET Statement 308

Chapter 7. Configuring the Site Table 309
Configuration Process 310
Step 1: Update the HOSTS.LOCAL Data Set 310
HOST Entries 310
NET and GATEWAY Entries 311
Sample HOSTS.LOCAL Data Set (HOSTS) 311
Step 2: Run MAKESITE 311
MAKESITE Command 313
TESTSITE Command 316

Chapter 1. Before You Begin

Before you begin to install or customize CS for OS/390, it is important that you:

- Review *OS/390 SecureWay Communications Server: IP Migration*.

  This book explains the differences between SecureWay Communications Server for OS/390 and the previous release and provides the information you need to plan the installation and configuration.
- Review "Overview for New Users" in this chapter.

  This table allows you to see at a glance the functions that SecureWay Communications Server for OS/390 offers you.
- Read "Configuration Data Sets and HFS Files" on page 13 in this chapter

  This information details the way in which different functions within TCP/IP search for the data sets and HFS files they use. A thorough understanding of this process will ensure that your system has access to the correct data sets or HFS files when they are needed.
- Have a copy of *OS/390 UNIX System Services Planning* handy.

  This book is referred to often.

When you have completed these steps, refer to "Customization Checklist" on page 28 to decide which chapters of this book to use. Your individual TCP/IP configuration might not require every server and function explained in this book. This checklist will guide you to the tasks and information you need.

Overview for New Users

Table 1 describes the functions TCP/IP provides and where you can find more information about them.

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390.

Function	Description	Additional Information
Connecting		

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
TCP/IP provides many connectivity options. **Note:** Each numbered item in the second column corresponds with a number in the third column.	1. ATM devices 2. Channel-to-channel (CTC) devices 3. HYPERchannel A220 Devices 4. LAN channel station (LCS) devices (IBM 8232 or IBM 3172) 5. MPCIPA devices 6. MPCOSA devices 7. MPCPTP devices 8. Common Link Access to Workstation (CLAW) devices (for example, RISC System/6000) 9. SNA LU0 Links 10. SNA LU 6.2 Links 11. X.25 NPSI Connections 12. Virtual Devices (VIPA) 13. 3745/46 Channel DLC Devices	1. "DEVICE and LINK Statement—ATM Devices" on page 148 2. "DEVICE and LINK Statement—CTC Devices" on page 155 3. "DEVICE and LINK Statement—HYPERchannel A220 Devices" on page 158 4. "DEVICE and LINK Statement—LAN Channel Station and OSA Devices" on page 161 5. "DEVICE and LINK Statement—MPCIPA Devices" on page 169 6. "DEVICE and LINK Statement—MPCOSA Devices" on page 172 7. "DEVICE and LINK Statement—MPCPTP Devices" on page 175 8. "DEVICE and LINK Statement—CLAW Devices" on page 151 9. "DEVICE and LINK Statement—SNA LU0 Links" on page 179 10. "DEVICE and LINK Statement—SNA LU 6.2 Links" on page 182 11. "DEVICE and LINK Statement—X.25 NPSI Connections" on page 185 12. "DEVICE and LINK Statement—Virtual Devices" on page 188 13. "DEVICE and LINK Statement—3745/46 Channel DLC Devices" on page 190
Logging in to remote hosts		
Telnet	Telnet is a terminal emulation protocol that allows you to log on to a remote host as though you are directly attached to that host. Telnet allows users on any host to have access to applications running on this host. You can establish and toggle between concurrent sessions with different hosts or multiple sessions with a single host.	"Chapter 11. Configuring the OS/390 UNIX Telnet Server" on page 387 "Chapter 12. Configuring the Telnet Server" on page 395
Transferring Files		
File Transfer Protocol (FTP)	File transfer through FTP makes the entire network a resource for users to store, share, and distribute data, regardless of the mix of devices and operating systems in the network. With file transfer, you can: • Manage remote host directories by listing, changing, creating, and deleting remote directories • Copy or move files and programs between your (local) host and remote hosts, or between two remote hosts.	"Chapter 14. Configuring the File Transfer Protocol (FTP) Server" on page 523

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
Running Programs on Remote Hosts		
Remote Execution (REXEC) Server and Client	With remote execution, you can send any command that is valid on a remote host and receive the results at the local host. Users can receive the results on their TSO terminals. This server runs the Remote EXecution Command Daemon (REXECD or OREXECD), which supports both the Remote Execution (REXEC) and Remote Shell (RSH) protocols. The REXECD and OREXECD servers handle commands issued by remote hosts. The servers perform automatic login and user authentication when user ID and password are entered.	"Chapter 17. Configuring the OS/390 Unix Remote Execution Server" on page 621 "Chapter 18. Configuring the Remote Execution Server" on page 625
Exchanging Mail Electronically		
sendmail	sendmail enables electronic mail across the internet. It provides support for SMTP and POP/3 clients.	"Chapter 20. Configuring OS/390 UNIX sendmail and Popper" on page 701
Simple Mail Transfer Protocol (SMTP)	The Simple Mail Transfer Protocol server sends electronic mail. It provides message and note exchange between TCP/IP hosts, but has no support for document translation. Mail is sent from the normal local mail application to a background SMTP application, which stores the mail in its own storage. The SMTP application is based on end-to-end delivery: it directly contacts the destination host's SMTP and keeps the mail in its local spool until it has been successfully copied to the recipient's SMTP.	"Chapter 19. Configuring the SMTP Server" on page 635
Printing files on remote printers		
Remote Print Server (LPD)	You can access a variety of remote printers to print, redirect, query, and remove jobs, using the Line Printer Client (LPR) protocol on the client host and the Line Printer Daemon (LPD) protocol on the print server. The TCP/IP Line Printer Requester (LPR client) on the host enables host users to send data to be printed on Line Printer Daemon (LPD) print servers on the TCP/IP network. LPD prints a local file on a printer accessed through a network. The line printer requestor (LPR) function and the line printer server (LPD) function are part of TCP/IP for MVS. Line printer daemon (LPD) is the printer server that allows hosts in your TCP/IP network to use printers on your MVS system. A client from any TCP/IP host can use the local line printer requestor command (LPR) to print a local file on a JES controlled printer.	"Chapter 24. Configuring the Remote Print Server (LPD)" on page 893

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
Network Print Facility (NPF)	NPF accepts print data from normal MVS sources and routes it to TCP/IP-supported printers. NPF emulates VTAM and JES destinations and sends the data to the Line Printer Daemon (LPD) via the Line Printer Client (LPR) using a routing file or table. It also provides user exits to allow system administrators to substitute their own routing mechanism to suit their particular installation requirements. You can print from anywhere on your network to your MVS printers.	*OS/390 eNetwork Communications Server: IP Network Print Facility*
Accessing mainframe relational databases		
Network Database (NDB)	NDB enables users on TCP/IP hosts to access DB2 resources in an interactive mode. The connection between the client and the NDB server in MVS is based on TCP protocols. For example, users of a UNIX or OS/2 workstation can use NDB to query or write MVS DB2 databases.	"Chapter 32. Configuring the Network Database (NDB) System" on page 1077
Managing Your Network		
Simple Network Management Protocol (SNMP) Server and Client	SNMP allows you to manage your TCP/IP network. SNMP provides functions for network monitoring and management. As the name implies, it is a simple protocol, minimizing the number and complexity of network management functions. SNMP is comprised of two components: a server and a client. The server is called an agent, and the client is called a manager. CS for OS/390 supports both the SNMP manager and agent functions.	"Chapter 22. Configuring Simple Network Management Protocol (SNMP)" on page 823
Naming hosts with the BIND-based Domain Name System (DNS)	DNS is like a telephone book that contains a person's name, address, and telephone number. The name server maps a host name to an IP address, or an IP address to a host name. For each host, the name server can contain IP addresses, nicknames, mailing information, and available well-known services such as SMTP, FTP, and Telnet. DNS now includes support for load balancing in a sysplex environment and also for dynamic IP.	"Chapter 21. Configuring the Bind-Based Domain Name System (DNS)" on page 709

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
Querying hosts and users in the network	TCP/IP provides a number of commands for displaying information about various networks, users, and your local host. For example, you can use the ping command to determine whether your workstation can connect to a particular host and the approximate time it takes for that host to respond. You can use the netstat command to display information about your local host, such as status of the TCP connections and devices. You can also use the DISPLAY TCPIP,,NETSTAT command to request netstat information.	*OS/390 SecureWay Communications Server: IP User's Guide*
Maintaining routing tables with OROUTED	To dynamically create and maintain network routing tables, you can use the OROUTED server. The OROUTED server uses the RIP protocol (version 1 and 2) to allow gateways and routers to create and maintain network routing tables. The OROUTED server determines whether a route is unavailable, or whether a more efficient route exists, and updates hosts' routing tables automatically. Virtual IP addressing allows inbound or outbound traffic to be routed around interface outages in a host provided that there is an alternate interface available.	"Chapter 25. Configuring the OROUTED Server" on page 917
Maintaining the host routing table with OMPROUTE	OMPROUTE provides an alternative to the static TCP/IP gateway definitions. When configured properly, the MVS host running with OMPROUTE becomes an active OSPF or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be used to dynamically maintain the host routing table.	"Chapter 26. Configuring OMPROUTE" on page 949 "Chapter 27. Using OSPF" on page 1021
Making NCP an active router with NCPROUTE server	The NCPROUTE server makes NCP an active router on a TCP/IP network and enables NCP to be responsive to SNMP route table queries. The NCPROUTE server maintains the routing tables on behalf of the NCP and response to routing queries. The tables are periodically distributed to the NCP, so it can perform its IP routing correctly.	"Chapter 28. Configuring the NCPROUTE Server" on page 1035
Registering the location of RPC applications with Portmapper	The Portmapper function registers the location of RPC applications. The PORTMAP address space keeps track of the relationship between RPC program numbers and the ports they are connected to.	"Chapter 29. Configuring the OS/390 Unix PORTMAP Address Space" on page 1065 "Chapter 30. Configuring the PORTMAP Address Space" on page 1067
Securing your network		

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
Kerberos server	TCP/IP is designed to verify authorization for a particular client/server function at the user level to protect sensitive data. The Kerberos server verifies authorization for client/server function at the user level. The Kerberos server includes an encryption facility that securely authenticates the application-client/server pair. To communicate with the Kerberos server, the client/server application pair must use the Kerberos libraries and the Kerberos ticket-granting services. Kerberos is an API that allows the programmer to force additional security checks before the client can use a server. Both server and client applications must have these APIs implemented.	"Chapter 23. Configuring the Kerberos Server" on page 871
Extending the functions of TCP/IP through programming interfaces		
Application programming interfaces, distributed programming interfaces, programming tools **Note:** Each numbered item in the second column corresponds with a number in the third column.	1. Berkeley Software Distribution (BSD**) socket interface 2. CICS socket interface 3. Sockets Extended 4. IMS socket interface API 5. Network Computing System (NCS) and Remote Procedure Call (RPC) 6. OSF/Motif widgets and header files 7. Simple Network Management Protocol distributed programming interface 8. X Window System API 9. X/Open transport interface (XTI) 10. X/Open Transport Interface (XTI 11. REXX sockets 12. Pascal interface	1. *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide* 2. *OS/390 SecureWay Communications Server: IP CICS Sockets Guide* 3. *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide* 4. *OS/390 eNetwork Communications Server: IP IMS Sockets Guide* 5. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 6. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 7. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 8. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 9. *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide* 10. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 11. *OS/390 SecureWay Communications Server: IP Programmer's Reference* 12. *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide*
Increasing productivity		

Table 1. Overview of Functions Available in SecureWay Communications Server for OS/390. (continued)

Function	Description	Additional Information
Testing and debugging applications with Miscellaneous Server (MISCSERV)	MISCSERV is a server that can be used to test and debug applications. It supports the Echo (RFC 862), Discard (RFC 863) and the Character Generator (RFC 864) protocols. The Echo protocol provides an echo server that sends back to the originating application any data it receives. The Discard protocol provides a discard service that throws away any data it receives and no response is sent back. The Character Generator protocol provides a service that sends a repetitive stream of character data without regard to its content.	"Chapter 33. Configuring the Miscellaneous (MISC) Server" on page 1091
Network Station Computer Support (thin client applications)		
Trivial FTP, DHCP, and TIMED	The TFTP server enables the transferring of files to and from a remote server. DHCP allows clients to obtain IP network configuration information, including an IP address, from a central DHCP server. TIMED is a TCP/IP daemon that is used to provide the date and time.	*Network Station Manager for S/390*
Managing Quality of Service for Your Network		
Policy Agent	The Policy Agent is a server that allows you to define and manage service policies. Service policies are used to define filters to select different types of traffic and to define the network service that the selected traffic will receive.	"Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent" on page 1095
RSVP Agent	The RSVP Agent is a server that allows for end-to-end resource reservation on behalf of applications. Applications request RSVP services via the RSVP API (RAPI).	"Chapter 35. Configuring the RSVP Agent" on page 1121
SLA Subagent	The SLA Subagent provides information about service policies and performance data about application mapped to those policies via the slapmPolicyStatsTable and the slapmSubcomponentTable.	"Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent" on page 1095

Configuration Data Sets and HFS Files

The following topics are discussed in this section:

- "Overview of Data Sets and HFS Files" on page 14
- "How CS for OS/390 Searches for Configuration Files" on page 14
- "Information Specific to Data Sets in Configuration File Search Orders" on page 14
- "Search Order and Configuration Files for the TCP/IP Stack" on page 16
- "Search Order and Configuration Files for TCP/IP Applications" on page 19
- "MVS System Symbols" on page 26

Overview of Data Sets and HFS Files

Data set and *file* are comparable terms. If you are familiar with MVS™, you probably use the term data set to describe a unit of data storage. If you are familiar with AIX or UNIX, you probably use the term file to describe a named set of records stored or processed as a unit. In the CS for OS/390 environment, in addition to the traditional MVS data set organizations (such as sequential, partitioned) the OS/390 UNIX files are arranged in a hierarchical file system (HFS) and are called HFS files.

Some data sets and HFS files have special importance because of their function. For example, certain data sets and HFS files are used when configuring the CS for OS/390 environment. Other data sets, like the Telnet server (Telnet daemon) perform specific communication functions. This section describes the data sets and HFS files necessary for configuring the CS for OS/390 environment and the search orders used to find them. A search order can include both HFS files and data sets, and these data sets and HFS files will be collectively referred to as the *configuration files* in this section.

How CS for OS/390 Searches for Configuration Files

It is important to understand how the CS for OS/390 environment searches for the configuration files, and when you can override the default search order with environment variables, JCL, or other variables you provide. This knowledge allows you to accommodate your local data set and HFS file naming standards, and it is helpful to know the data set or HFS file in use when diagnosing problems.

An important point to note is that what is being referred to here as the CS for OS/390 environment consists of the CS for OS/390 stack, the CS for OS/390 applications (OS/390 UNIX System Services applications, or those written to the OS/390 UNIX sockets API) and TCP/IP MVS applications (those written to the TCP/IP MVS APIs [C, Sockets Extended, CICS®, IMS™, REXX]). The TCP/IP stack and both sets of applications have some common (or global) configuration files, but they also use configuration files that are different. The next section provides data set specific information, and the following two sections look at the global configuration files and search orders for each of them.

Another important point to note is that when a search order is applied for any configuration file, the search ends with the first file found. Therefore, unexpected results are possible if you place configuration information in a file that never gets found, either because other files exist earlier in the search order, or because the file is not included in the search order chosen.

Information Specific to Data Sets in Configuration File Search Orders

Within the search order for the configuration files, there are data sets listed that are explicitly or dynamically (implicitly) allocated (the term allocation refers to the process of requesting access to a data set).

Explicit Data Set Allocation

Explicitly allocated data sets in a configuration file search order consist of those data sets that you specify through the use of DD statements in JCL procedures.

Dynamic Data Set Allocation

CS for OS/390 makes extensive use of dynamically allocated data sets using the MVS Dynamic Data Set Allocation function. Multiple versions of a data set can exist, each having a different high-level qualifier or middle-level qualifier. The search order for any configuration file will determine which data set is found and used.

High-Level Qualifier: TCP/IP is distributed with a default high-level qualifier (HLQ) of *TCP/IP*. This HLQ is a hard-coded character string within CS for OS/390.

For dynamic data set allocation, you can accept the default HLQ or override it. To override the default HLQ used by dynamic data set allocation, specify the DATASETPREFIX statement in the TCPIP.DATA configuration file. The DATASETPREFIX value is used as the last step in the search order for most configuration files. Note that DATASETPREFIX is not used as the last step in the search order for the PROFILE.TCPIP and TCPIP.DATA configuration files.

Middle-Level Qualifiers: Multiple middle level qualifiers (MLQ) permit the isolation of certain profile and translation table data sets. Two of the possible middle-level qualifiers are:

- Node name

  Node name is a MLQ used in the search order for finding the configuration file PROFILE.TCPIP. Node name is determined by the parameters specified during VMCF initialization. For further information on initializing VMCF, see the *SecureWay Communications Server for OS/390 Program Directory*.
- Function name

  The CS for OS/390 implementation of national language support (NLS) and double-byte character set (DBCS) support requires the use of multiple translation tables. To facilitate the concurrent use of multiple languages and code pages, TCP/IP uses a middle-level qualifier to designate which server or client uses a particular translation table. STANDARD, the default MLQ, is available for use if a single translation table can be used by multiple servers or clients. The TCP/IP commands TELNET client and FTP provide a TRANSLATE parameter that permits you to specify your chosen MLQ to replace the function name for that invocation of the command. For example, SRVRFTP is used as a MLQ by the File Transfer Protocol server.

Dynamically Allocated Data Sets: A dynamically allocated data set can have from 1 to 6 different fully-qualified data set names, depending on the function of the data set. The general naming convention and its specific application are explained in “Naming Conventions for Dynamically Allocated Data Sets” on page 16.

Following are some of the data sets that can be dynamically allocated by TCP/IP in a configuration file search order (you cannot specify them with DD statements in JCL):

```
ETC.PROTO               ETC.RPC
HOSTS.ADDRINFO          HOSTS.SITEINFO
SRVRFTP.TCPCHBIN        SRVRFTP.TCPHGBIN
SRVRFTP.TCPKJBIN        SRVRFTP.TCPSCBIN
SRVRFTP.TCPXLBIN        STANDARD.TCPCHBIN
STANDARD.TCPHGBIN       STANDARD.TCPKJBIN
STANDARD.TCPSCBIN       STANDARD.TCPXLBIN
```

For each of these data sets, the fully-qualified name is established by using one of the following values as the data set HLQ:

- User ID or job name

- DATASETPREFIX value

Naming Conventions for Dynamically Allocated Data Sets: A data set that you create for the purpose of being allocated dynamically by TCP/IP must use the following naming conventions. A data set that you allocate explicitly (with a DD statement in JCL) can have any valid MVS data set name or HFS file name.

Note: In the examples below, xxxx indicates an appropriate high-level qualifier, yyyy indicates an appropriate middle-level qualifier, and zzzz indicates an appropriate low-level qualifier.

- *userid.yyyy.zzzz*

  userid is the user ID of the logged on TSO user.
- *TSOprefix.yyyy.zzzz*

  TSOprefix is the data set prefix established by the TSO PROFILE command. *userid* is the default value of *TSOprefix*
- *jobname.yyyy.zzzz*

  jobname is the job name specified on the JOB statement for a job stream or the procedure name for a started procedure.
- *hlq.yyyy.zzzz*

  hlq is the TCP/IP HLQ distributed as the system default, which can be overridden by the value in the DATASETPREFIX statement.
- *xxxx.nodename.zzzz*

  nodename is an MLQ that is used to define the data set name for the TCP/IP stack profile data set.
- *xxxx.function_name.zzzz*

  function_name denotes an acronym specifying a particular TCP/IP server (for example SRVRFTP for the FTP server) and is used as a MLQ for the translation table data set for that application.
- *xxxx.private_name.zzzz*

  private_name is a user-specified private qualifier that can be specified as an option on some TCP/IP commands.
- SYS1.TCPPARMS(TCPDATA)

  member of a system data set used to find the *configuration file* TCPIP.DATA. (You can allocate the SYS1.TCPPARMS data set with partitioned organization (PO), a fixed block format (FB), a logical record length of 80, and any valid blocksize for a fixed block, such as 3120.)

Search Order and Configuration Files for the TCP/IP Stack

Two configuration files are used by the TCP/IP stack, PROFILE.TCPIP and TCPIP.DATA. PROFILE.TCPIP is used only for the configuration of the TCP/IP stack. TCPIP.DATA is used during configuration of both the TCP/IP stack and applications; the search order used to find TCPIP.DATA is the same for both the TCP/IP stack and applications.

See“Appendix G. Protocol Number and Port Assignments” on page 1193 for more information about protocol-number assignments and test configuration port assignments used in the PROFILE.TCPIP configuration data set and the HFS file /etc/services.

PROFILE.TCPIP Search Order

During initialization of the TCP/IP stack (also referred to in this book as the TCP/IP address space), system operation and configuration parameters for the TCP/IP stack are read from the configuration file PROFILE.TCPIP. The search order used by the TCP/IP stack to find PROFILE.TCPIP involves both explicit and dynamic data set allocation as follows:

- //PROFILE DD DSN=*aaa.bbb.ccc(anyname)*

  Explicitly specifying the PROFILE DD statement in the TCPIPROC JCL is the recommended way to specify PROFILE.TCPIP. If this DD statement is present, the data set it defines is explicitly allocated by MVS. No dynamic allocation is done. The rest of the data sets looked for in the PROFILE.TCPIP search order are dynamically allocated by TCP/IP.
- *jobname.nodename*.TCPIP
- *hlq.nodename*.TCPIP
- *jobname*.PROFILE.TCPIP
- *hlq*.PROFILE.TCPIP

Remember that for the above PROFILE.TCPIP search order, the default *hlq* distributed with TCP/IP is the string *TCPIP*. For detailed information about the content and specification of the configuration file PROFILE.TCPIP, see “Chapter 4. Configuring the TCP/IP Address Space” on page 97.

Examples: The following examples show the search order used by TCP/IP to find the configuration file PROFILE.TCPIP.

Search Order When DD Cards Are In Your TCP/IP Startup Procedure: In this example, the PROFILE and SYSTCPD DD cards are specified as follows:

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5647-A01 (C) Copr. IBM Corp. 1991,1998.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=7500K,TIME=1440
//*
:
:

//PROFILE DD DISP=SHR,DSN=MVSA.PROD.PARMS(PROFILE)
:
:
```

Because the PROFILE DD is the first step in the search order, TCP/IP uses the data set MVSA.PROD.PARMS(PROFILE) as the PROFILE.TCPIP configuration file.

Search Order When No DD Cards Are In Your TCP/IP Startup Procedure: For this example, the TCPIPROC JCL does not have the PROFILE card.

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5647-A01 (C) Copr. IBM Corp. 1991,1998.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=7500K,TIME=1440
//*
:
```

For the configuration file PROFILE.TCPIP, the search order used is as follows:

1. PROFILE DD

   No PROFILE DD exists.
2. *jobname.nodename.*TCPIP

   If *jobname..nodename.*TCPIP is found, the search stops here.
3. *hlq.nodename.*TCPIP

   If *hlq..nodename.*TCPIP is found, the search stops here.
4. *jobname.*PROFILE.TCPIP

   If *jobname.*.PROFILE.TCPIP is found, the search stops here.
5. *hlq.*PROFILE.TCPIP

   hlq..PROFILE.TCPIP is searched last if necessary. Remember that the default *hlq* distributed with TCP/IP is the string *TCPIP*. So, effectively, this is TCPIP.PROFILE.TCPIP in the search order.

TCPIP.DATA Search Order

During initialization of the TCP/IP stack, the configuration file TCPIP.DATA is also used. The search order used to find TCPIP.DATA is as follows:

- The MVS data set or HFS file that is identified in an environment variable called RESOLVER_CONFIG.

  This environment variable is passed as a parameter to the TCP/IP stack in the TCPIPROC JCL used to start TCP/IP. The following is an example of specifying this environment variable in the JCL:

```
//TCPIP    PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//*  SMP/E Distribution Name: EZAEB01G
//*
//*  Licensed Materials - Property of IBM
//*  "Restricted Materials of IBM"
//*  5647-A01
//*  (C) Copyright IBM Corp. 1991, 1999
//*  Status = CSV2R8
//*
//TCPIP    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//         PARM=('&PARMS',
//        'ENVAR("RESOLVER_CONFIG=//''TCPIVP.TCPPARMS(TCPDATA)''")')
//*
```

- /etc/resolv.conf

  This is the file /etc/resolv.conf that resides in the HFS.

- //SYSTCPD DD DSN=*ddd.eee.fff(anyname)*

  The //SYSTCPD DD card can be specified in TCPIPROC JCL.
- *userid*.TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
- SYS1.TCPPARMS(TCPDATA)

  For more information on this data set, see “Naming Conventions for Dynamically Allocated Data Sets” on page 16.
- *hlq*.TCPIP.DATA

  Remember that the default *hlq* distributed with TCP/IP is the string *TCPIP*. So, effectively, this is TCPIP.TCPIP.DATA in the search order.

If the SYSTCPD DD is used, remember that it is only searched for if the HFS file /etc/resolv.conf does not exist. For more information about a multiple TCP/IP instance environment, see “Considerations for Multiple Instances of TCP/IP” on page 57.

Note: The Telnet hostname mapping function does not use the RESOLVER_CONFIG TCPIP.DATA file nor will it use the default HFS TCPIP.DATA file. If Telnet hostname mapping is used, the SYSTCPD DD or other default must be used to specify the resolver search order. If RESOLVER_CONFIG specifies an MVS data set, the SYSTCPD DD should specify the same MVS data set.

Search Order and Configuration Files for TCP/IP Applications

This section describes the configuration files used in common (that is, globally) by the applications, and the search orders for those configuration files. Each application can also have its own configuration files that are specific to that application. For information about specific configuration files, see the descriptions of the individual applications in “Part 2. Configuring the Servers” on page 317. Some examples of these specific files appear in the following text.

Resolver Configuration Files

As shown in Figure 1 on page 20, files called “resolver configuration files” are used by the resolver component, which is part of the socket library. TCP/IP MVS sockets deliver one resolver that is used with TCP/IP MVS APIs (C, Sockets Extended, CICS, IMS, REXX). Language Environment for OS/390 V2R5 and later releases delivers another resolver that is used by all OS/390 UNIX socket programs.

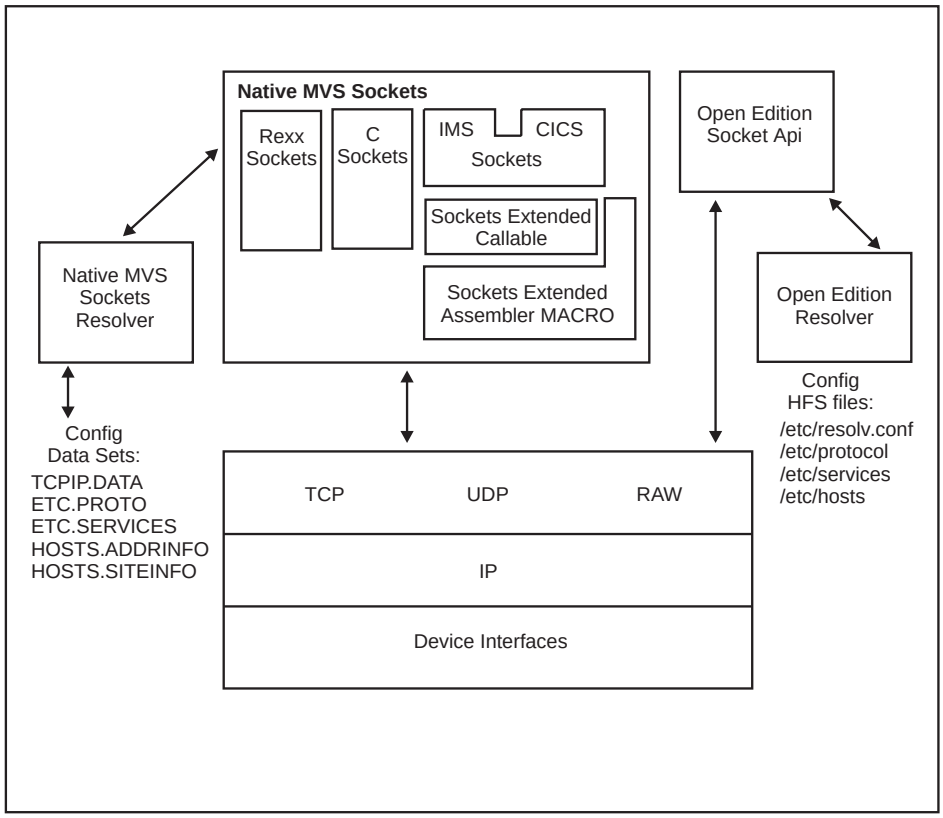


Figure 1. Resolver Components and Related Configuration Files in OS/390 UNIX and TCP/IP Environments

Not only is there a search order for each type of data set, for each data set or file there is a hierarchy of possible locations.

For the TCP/IP resolver, the TCPIP.DATA data set has the following search order:

1. Any data set that is allocated to DDname SYSTCPD
2. jobname.TCPIP.DATA for batch jobs, or userid.TCPIP.DATA for TSO users
3. SYS1.TCPPARMS(TCPDATA)
4. TCPIP.TCPIP.DATA

For the Language Environment resolver, hereafter called the LE resolver, the base resolver configuration data set or file (the one that corresponds to TCPIP.DATA) has the following search order:

1. The MVS data set or HFS file that is identified in an environment variable called RESOLVER_CONFIG

   If the environment variable RESOLVER_CONFIG has been defined, the resolver uses the value of this environment variable as the name of an MVS data set or HFS file to access the resolver configuration data. The syntax for an MVS data set name is “//’mvs.dataset.name’”. The syntax for an HFS file name is “/dir/subdir/file.name”.

   Note: This step will fail if the data set or file is not available or has been allocated exclusively elsewhere.

2. /etc/resolv.conf
3. Any data set that is allocated to DD-name SYSTCPD

   This option may not be a good technique for processes that use the fork() command. This allocation will not be available to the child process that is forked because DD allocations for the parent process are not inherited by the child. The only exception to this rule is a STEPLIB allocation.
4. userid.TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
5. SYS1.TCPPARMS(TCPDATA)
6. TCPIP.TCPIP.DATA

Similar hierarchies exist for all the resolver configuration data sets or files as shown in the following table:

Table 2. Resolver Configuration Data Sets and Files

TCP/IP Resolver	**LE Resolver**
Base resolver configuration:	**Base resolver configuration**:
1. SYSTCPD DD-name 2. jobname.TCPIP.DATA or userid.TCPIP.DATA 3. SYS1.TCPPARMS(TCPDATA) 4. TCPIP.TCPIP.DATA	1. RESOLVER_CONFIG environment variable 2. /etc/resolv.conf 3. SYSTCPD DD-name 4. userid.TCPIP.DATA 5. SYS1.TCPPARMS(TCPDATA) 6. TCPIP.TCPIP.DATA
Local hosts tables:	**Local hosts tables**:
1. jobname.HOSTS.xxxxINFO or userid.HOSTS.xxxxINFO 2. *hlq*.HOSTS.xxxxINFO	1. X_SITE and X_ADDR environment variable 2. /etc/hosts 3. userid.HOSTS.xxxxINFO 4. *hlq*.HOSTS.xxxxINFO
Protocol information:	**Protocol information**:
1. jobname.ETC.PROTO or userid.ETC.PROTO 2. *hlq*.ETC.PROTO	1. /etc/protocol 2. userid.ETC.PROTO 3. *hlq*.ETC.PROTO
Service information:	**Service information**:
1. SERVICES DD-name 2. jobname.ETC.SERVICES or userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES	1. /etc/services 2. userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES
Translate table:	**Translate table**:
1. jobname.STANDARD.TCPXLBIN or userid.STANDARD.TCPXLBIN 2. *hlq*.STANDARD.TCPXLBIN	1. X_XLATE environment variable 2. *hlq*.STANDARD.TCPXLBIN 3. Internal hard-coded translate table
Note: *hlq.* comes from the DATASETPREFIX parameter in the base resolver configuration data set (TCPIP.DATA)	**Note:** *hlq.* comes from the DATASETPREFIX parameter in the base resolver configuration data set or file (TCPIP.DATA or /etc/resolv.conf Where environment variables are used, they can either refer to an MVS data set name or to an HFS file name.

Environment Variables: Some of the search orders described below include values of environment variables. Environment variables for the LE resolver include:

RESOLVER_CONFIG
The resolver configuration data sets or files.

X_SITE and X_ADDR
The HOSTS.SITEINFO and HOSTS.ADDRINFO data sets or files.

X_XLATE
The ASCII-EBCDIC translate table data set or file built by the TCP/IP for MVS CONVXLAT utility.

How to set an environment variable so that an OS/390 UNIX application is able to retrieve the value depends on whether the OS/390 UNIX application is started from the OS/390 shell or from JCL.

If the OS/390 UNIX application is to be started from the OS/390 shell, the *export* shell command can be used to set the environment variable. For example, to set the value of RESOLVER_CONFIG to the HFS file /etc/tcpa.data, you can code the following export command:

```
export RESOLVER_CONFIG=/etc/tcpa.data
```

If instead of an HFS file, you want to set RESOLVER_CONFIG to the data set MVSA.PROD.PARMS(TCPDATA) you can specify the following export command (be sure to put the single quotes around the data set name—if you do not, your user ID will be added as a prefix to the data set name when TCP/IP tries to open the file):

```
export RESOLVER_CONFIG="//'MVSA.PROD.PARMS(TCPDATA)'"
```

If the OS/390 UNIX application is to be started from JCL instead of from the OS/390 shell, the environment variable needs to be passed as a parameter in that OS/390 UNIX application's JCL. For example, following is an example that shows the RESOLVER_CONFIG variable set to pick up the TCPIP.DATA information from a file in the HFS:

```
//OSNMPD    PROC
//*
//* Procedure for running the OE SNMP agent
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/'
//      'ENVAR("RESOLVER_CONFIG=/etc/tcpa.data")/-d 0'))
:
```

The following example shows the RESOLVER_CONFIG variable set to pick up the TCPIP.DATA information from a partitioned data set:

```
//OSNMPD    PROC
//*
//* Procedure for running the OE SNMP agent
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/'
//  'ENVAR("RESOLVER_CONFIG=//''TCPA.MYFILE(TCPDATA)''")/-d 0'))
:
```

Understanding TCP/IP Data Set Names with OS/390 UNIX Services: The OS/390 UNIX socket runtime library (RTL) functions can use the following TCP/IP data sets:

- *tcpip.*TCPIP.DATA

- *tcpip*.STANDARD.TCPXLBIN
- *tcpip*.HOSTS.SITEINFO
- *tcpip*.HOSTS.ADDRINFO
- *tcpip*.ETC.PROTO
- *tcpip*.ETC.SERVICES

tcpip represents the value of the DATASETPREFIX in the TCPIP.DATA data set, if it was found; otherwise *tcpip* is by default TCP/IP.

Note: Beginning with Language Environment 1.7 for OE_SOCKETS and XOPEN_SOCKETS, initialization of the Resolver environment is deferred until the first request for the Internet Protocol (IP) Address Resolution function.

TCPIP.DATA

This file is referenced to determine, among other things, the data set prefix (DATASETPREFIX keyword) to be used when trying to access the rest of the configuration files specified in this section. For the search order used to find the TCPIP.DATA configuration file, see “Search Order and Configuration Files for TCP/IP Applications” on page 19.

STANDARD.TCPXLBIN

This file is referenced to determine the translate data sets to be used.

The search order used to access this configuration file is:

1. The value of the environment variable X_XLATE

   The value of the environment variable is used to fopen() the configuration file. All OS/390 UNIX I/O rules apply. For a complete discussion, see the chapter on ″Performing HFS I/O Operations″ in the *OS/390 C/C++ Programming Guide*.
2. *hlq*.STANDARD.TCPXLBIN

   hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); otherwise, *hlq* is TCPIP by default.

HOSTS.SITEINFO

This file supplies the information for the following four network host database functions: gethostbyname(), sethostent(), gethostent(), and endhostent. Additionally, it supplies information for the network database function getnetbyname().

The search order used to access this configuration file is:

1. The value of the environment variable X_SITE

   The value of the environment variable is used as is to fopen() the configuration file. All OS/390 UNIX I/O rules apply. For a complete discussion, see the chapter on ″Performing HFS I/O Operations″ in the *OS/390 C/C++ Programming Guide*.

   The only valid data set identified by this environment variable must contain the HOSTS.SITEINFO information created by the MAKESITE command.

   It is not recommended that an X_SITE refer to an HFS file, because the two types of data sets are incompatible.
2. /etc/hosts that resides in the HFS
3. *userid*.HOSTS.SITEINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).

4. *hlq*.HOSTS.SITEINFO

   hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); Otherwise, *hlq* is TCP/IP by default.

HOSTS.ADDRINFO

This file supplies the information for the following four network database functions: getnetbyaddr(), setnetent(), getnetent(), and endnetent(). Additionally, it supplies information for the network host database function gethostbyaddr().

The search order used to access this configuration file is:

1. The value of the environment variable X_ADDR

   The value of the environment variable is used as is to fopen() the configuration file. All OS/390 UNIX I/O rules apply. For a complete discussion, see the chapter on ″Performing HFS I/O Operations″ in the *OS/390 C/C++ Programming Guide*.

   The only valid data set identified by this environment variable must contain the HOSTS.ADDRINFO information created by the MAKESITE command.
2. /etc/hosts, only if the request was gethostbyaddr(); otherwise, this step is skipped. /etc/hosts resides in the HFS.
3. *userid*.HOSTS.ADDRINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
4. *hlq*.HOSTS.ADDRINFO

   hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); Otherwise, *hlq* is TCP/IP by default.

ETC.PROTO

This file supplies the information for the following five protocol database functions: getprotobynumber(), getprotobyname(), setprotoent(), getprotoent(), and endprotoent(). See “Protocol Assignments from /etc/protocol” on page 1193 for more information.

The search order used to access this configuration file is:

1. /etc/protocol that resides in the HFS
2. *userid*.ETC.PROTO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq.ETC.PROTO*

   hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); Otherwise, *hlq* is TCP/IP by default.

ETC.SERVICES

This file supplies the information for these five services database functions: getservbyport(), getservbyname(), setservent(), getservent(), and endservent(). See “/etc/services Port Assignments” on page 1194 for more information.

The search order used to access this configuration file is:

1. /etc/services that resides in the HFS
2. *userid*.ETC.SERVICES, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
3. hlq.ETC.SERVICES

hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); Otherwise, *hlq* is TCP/IP by default.

Examples

The following example shows the FTP server start procedure, which indirectly requests information from common (that is, global) configuration files TCPIP.DATA, HOSTS.SITEINFO, HOSTS.ADDRINFO, and ETC.SERVICES. The FTP server does not search for these files directly, but calls services that search for them. For this example, the EZAFTPAP JCL that is used is as follows:

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//*********************************************************************
//*                                                                   *
//*                                                                   *
//*                                                                   *
//*         Descriptive Name:            FTP Server Start Procedure   *
//*                                                                   *
//*         File Name:                   tcpip.SEZAINST(EZAFTPAP)     *
//*                                      tcpip.SEZAINST(FTPD)         *
//*                                                                   *
//*         SMP/E Distribution Name:     EZAFTPAP                     *
//*                                                                   *
//*                                                                   *
//*             Licensed Materials - Property of IBM                  *
//*             This product contains "Restricted Materials of IBM"   *
//*             5647-A01 (C) Copyright IBM Corp. 1995, 1997.          *
//*             All rights reserved.                                  *
//*             US Government Users Restricted Rights -               *
//*             Use, duplication or disclosure restricted by          *
//*             GSA ADP Schedule Contract with IBM Corp.              *
//*             See IBM Copyright Instructions.                       *
//*                                                                   *
//*                                                                   *
//*********************************************************************
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM='POSIX(ON) ALL31(ON)/&PARMS'
⋮

//*      SYSTCPD explicitly identifies which file is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the JCL of
//*      the server.  The file can be any sequential data set,
//*      member of a partitioned data set (PDS), or HFS file.
//*SYSTCPD DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
⋮
```

So, the search sequence for each configuration file is as follows:

- TCPIP.DATA
  1. Environment variable RESOLVER_CONFIG

     Because the FTP server is being started from JCL (rather than from the OS/390 shell), this environment variable would have to be passed as a parameter to FTPD. Because RESOLVER_CONFIG is not passed by this JCL, the search goes to the next step in the sequence.
  2. /etc/resolv.conf

     If this HFS file exists, the search stops here.
  3. The data set specified on the SYSTCPD DD card

     Because the SYSTCPD DD card is commented out in the above JCL, the search goes to the next step in the sequence.

4. *userid.*TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
5. SYS1.TCPPARMS(TCPDATA)

   If this data set is found, the search stops here.
6. *hlq.*TCPIP.DATA

   TCPIP.TCPIP.DATA is searched last, if necessary.

- HOSTS.SITEINFO
  1. The value of the environment variable X_SITE

     Because the X_SITE environment variable was not passed in the JCL, the search goes to the next step in the sequence.
  2. /etc/hosts that resides in the HFS
  3. *userid.*HOSTS.SITEINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
  4. *hlq.*HOSTS.SITEINFO

     The *hlq* used depends on the DATASETPREFIX statement found in the configuration file TCPIP.DATA. If no DATASETPREFIX statement is specified in TCPIP.DATA, the default value TCPIP is used and TCPIP.HOSTS.SITEINFO is the last file in the search order.
- HOSTS.ADDRINFO
  1. The value of the environment variable X_ADDR

     Because the X_ADDR environment variable was not passed in the JCL, the search goes to the next step in the sequence.
  2. /etc/hosts that resides in the HFS

     If this HFS file exists, the search stops here.
  3. *userid.*HOSTS.ADDRINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
  4. *hlq.*HOSTS.ADDRINFO

     The *hlq* used depends on the DATASETPREFIX statement found in the configuration file TCPIP.DATA. If no DATASETPREFIX statement is specified in TCPIP.DATA, the default value TCPIP is used and TCPIP.HOSTS.ADDRINFO is the last file in the search order.
- ETC.SERVICES
  1. /etc/services that resides in the HFS

     If this HFS file exists, the search stops here.
  2. *userid.*ETC.SERVICES, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
  3. *hlq.*ETC.SERVICES

     The *hlq* used depends on the DATASETPREFIX statement found in the configuration file TCPIP.DATA. If no DATASETPREFIX statement is specified in TCPIP.DATA, the default value TCP/IP is used and TCPIP.ETC.SERVICES is the last file in the search order.

MVS System Symbols

Use of MVS system symbols in the PROFILE.TCPIP and OBEYFILE data sets is automatically supported. For MVS system symbols in other configuration files, such as TCPIP.DATA, use the symbol translator utility, EZACFSM1, to translate the symbols before the files are read by TCP/IP. EZACFSM1 reads an input file and writes to an output file, translating any symbols in the process. The input file and

output file can be MVS data sets or HFS files, but do not specify the same file for both the input and output files (this results in a return code of 45 and no translation is attempted).

Following is the symbol translator JCL, found in *hlq*.SEZAINST(CONVSYM), which is used to start EZACFSM1:

```
//________ JOB (accounting,information),programmer.name,
//             MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A
//*
//*   CS for OS/390 IP
//*   SMP/E distribution name: EZACFCSY
//*
//*   5647-A01 (C) Copyright IBM Corp. 1998.
//*   Licensed Materials - Property of IBM
//*
//*   Function: System Symbols Translator JCL
//*
//*   This JCL kicks off a utility that will read from
//*   an input file that contains MVS System Symbols
//*   and produce an output file which has those symbols
//*   replaced with their substitution text, as defined
//*   in the appropriate IEASYMxx PARMLIB data set; see MVS
//*   Initializaton and Tuning Reference for rules about symbols.
//*
//*   This JCL can be run against any of the TCP/IP configuration
//*   files that contain MVS System Symbols.  An example of how it
//*   could be used is this; a customer could have one base TCPIP.DATA
//*   file containing MVS System Symbols which they edit and maintain.
//*   They would run this utility against this one file the various
//*   MVS systems to produce the TCPIP.DATA file for each different
//*   system.
//*
//STEP1   EXEC  PGM=EZACFSM1,REGION=0K
//SYSIN   DD DSN=TCP.DATA.INPUT,DISP=SHR
//*SYSIN   DD    PATH='/tmp/tcp.data.input'
//*          The input file can be either an MVS file or an HFS file.
//*
//*
//SYSOUT  DD DSN=TCP.DATA.OUTPUT,DISP=SHR
//*SYSOUT  DD    PATH='/tmp/tcp.data.output',PATHOPTS=(OWRONLY,OCREAT),
//*              PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*          The output file can be either an MVS file or an HFS file.
//*
//*          The output file cannot be the same file as the input file-
//*          doing so will result in a return code of 45.
//*
//*          You can mix input and output file types (i.e., the input
//*          can be an MVS file with the output an HFS file, or visa
//*          versa).
//*          Note: Other pathmodes for sysout may be used if needed.
```

The symbol translator utility can be used on any of the TCP/IP configuration files, but because the PROFILE.TCPIP file is automatically translated during TCP/IP initialization, there is no need to run the utility against that file.

Changes to the Native TCP/IP Environment

If you are not currently converting to OS/390 UNIX applications and you prefer to keep a TCP/IP MVS appearance in your day-to-day operations, some basic changes have been made which might result in changes to your configuration. Some of the TCP/IP MVS applications shipped in previous releases have been re-written as OS/390 UNIX applications. These include the following:

- FTP client and server—see “Chapter 14. Configuring the File Transfer Protocol (FTP) Server” on page 523.
- OROUTED—see “Chapter 25. Configuring the OROUTED Server” on page 917.
- DNS—see “Chapter 21. Configuring the Bind-Based Domain Name System (DNS)” on page 709.
- SNMP—see “Chapter 22. Configuring Simple Network Management Protocol (SNMP)” on page 823.

Information on these applications can also be found in the *OS/390 SecureWay Communications Server: IP Migration* and the *OS/390 SecureWay Communications Server: IP User's Guide*.

When a sockets application uses OS/390 UNIX services, there are some configuration related consequences:

- OS/390 UNIX applications will use the LE resolver (see Figure 1 on page 20), which will use the OS/390 UNIX search order (see Table 2 on page 21) to find the datasetprefix (or hlq) for accessing files, looking up host names and addresses, local port numbers, and protocol numbers.
- OS/390 UNIX applications will most likely have enhanced the search order for specific client and server configuration files to include HFS files.

Using Automatic Restart Manager

Automatic restart manager is an OS/390 function that can automatically restart the TCP/IP stack after an abnormal end (abend).

During initialization, TCP/IP automatically registers with the automatic restart manager, using the following options:

```
REQUEST=REGISTER
ELEMENT=EZAsysclonetcpname
```

where:

- sysclone is one or two character shorthand notation for the name of the MVS system. Refer to *OS/390 MVS Initialization and Tuning Guide* for a complete description of the SYSCLONE static system symbol.
- tcpname is a one to eight character name of the TCP/IP stack which registers with the automatic restart manager. For example, if the SYSCLONE value is 02 and the TCP/IP stack name is TCPCS, the resulting ELEMENT value is EZA02TCPCS.

```
ELEMTYPE=SYSTCPIP
TERMTYPE=ELEMTERM
```

For more information about automatic restart manager, refer to *OS/390 MVS Setting Up a Sysplex*.

Customization Checklist

Your individual TCP/IP configuration might not have every server and function explained in this book. Use the following checklist to decide which tasks you have to do to customize your system and which chapters of this book to use. It shows when each task is required, where in the book it is explained, and which sample data sets and procedures it uses. You can find all the samples in *hlq*.SEZAINST or in the /usr/lpp/tcpip/samples directory.

Table 3. Checklist of Customization Tasks

Req	When	Task and Reference	Sample Data Sets	Sample Procedures
✓	Always	Planning Customization of CS for OS/390 Chapter 2. on page 35	VTAMLST	
✓	Always	Configuring the TCPIP Address Space Chapter 4. on page 97	SAMPPROF	TCPIPROC
✓	Always	Defining the TCP/IP Client System Parameters Chapter 6. on page 285	TCPDATA	
✓	Always	Configuring the Site Table Chapter 7. on page 309	HOSTS	
	If your network connects TCP/IP hosts using the LU0 protocol over an SNA backbone	Configuring the SNALINK Environment Chapter 8. on page 333	VTAMLST	SNALPROC
	If your network connects TCP/IP hosts using the LU6.2 protocol over an SNA backbone	Configuring the SNALINK LU6.2 Interface Chapter 9. on page 347	VTAMLST LU62CFG	LU62PROC
	If your network will be communicating (sending and receiving IP datagrams) with a Packet Switching Data Network	Configuring the X.25 NPSI Server Chapter 10. on page 361	VTAMLST NPSIDATE NPSIGATE	X25PROC
	If you want to use Telnet to allow users on any host to have access to applications running on this host	Configuring the OS/390 UNIX Telnet Server Chapter 11. on page 387		
	If you will be configuring the Telnet server.	Configuring the Telnet Server Chapter 12. on page 395	VTAMLST SAMPPROF	TCPIPROC
	If you will be using the File Transfer Program (FTP) Server to send or receive files across the network	Configuring the File Transfer Program (FTP) Server Chapter 14. on page 523	FTCDATA FTPSDATA	EZAFTPAP
	If you will be using the Remote Execution Server to execute TSO commands that have been received from a remote host	Configuring the OS/390 UNIX Remote Execution Server Chapter 17. on page 621		If you will be configuring the remote execution server.
		Configuring the Remote Execution Server Chapter 18. on page 625		RXPROC
	If you will be using the Simple Mail Transfer Protocol server to send electronic mail	Configuring the SMTP Server Chapter 19. on page 635	SMTPNOTE SMTPCONF	SMTPPROC

Table 3. Checklist of Customization Tasks (continued)

Req	When	Task and Reference	Sample Data Sets	Sample Procedures
	If you will be using the Domain Name Server to map domain names to IP addresses	Configuring the Domain Name Server Chapter 21. on page 709		
	If you will be using Simple Network Management Protocol (SNMP) to manage TCP/IP agents in the network	Configuring Simple Network Management Protocol Chapter 22. on page 823	MIBDESC mibs.data osnmpd.data snmpd.conf snmpv2.conf	OSNMPDPR SNMPPROC
	If you will be using the Kerberos server to verify authorization for client/server function at the user level	Configuring the Kerberos Server Chapter 23. on page 871	SERVICES KRBCONF ADMADD ADMGET ADMMOD	MVSKERB ADM@SERV
	If you will be using the Remote Print Server (LPD) to print a local file on a printer accessed through a network	Configuring the Remote Print Server (LPD) Chapter 24. on page 893	LPDDATA	LPSPROC
	If you will be using the OROUTED server to use the RIP protocol to allow gateways and routers to create and maintain network routing tables	Configuring the OROUTED Server Chapter 25. on page 917	SERVICES EZARTPRF EZARTGW	OROUTED
	If you will be using the OMPROUTE application for OSPF or RIP routing	Configuring OMPROUTE Chapter 26. on page 949		OMPROUTE
	If you will be using OSPF.	Using OSPF Chapter 27. on page 1021		
	If you will be using the NCPROUTE server to make NCP an active router on a TCP/IP network and to enable NCP to be responsive to SNMP route table queries	Configuring the NCPROUTE Server Chapter 28. on page 1035	NCPRPROF SERVICES EZBNRPRF EZBNRGW	NCPROUT
	If you will be using the Portmapper function to register the location of RPC applications	Configuring the OS/390 UNIX PORTMAP Address Space Chapter 29. on page 1065		OPORTPRC
	If you will configuring the PORTMAP address space.	Configuring the PORTMAP Address Space Chapter 30. on page 1067		PORTPROC
	If you will be using the Network Computing System (NCS) servers	Configuring the NCS Interface Chapter 31. on page 1073		NRGLBD LLBD
	If you will be using the NDB server to gain access to databases on a mainframe	Configuring the Network Database (NDB) System Chapter 32. on page 1077	ETCRPC	PORTSPRC PORTCPRC

Table 3. Checklist of Customization Tasks (continued)

Req	When	Task and Reference	Sample Data Sets	Sample Procedures
	If you will be using the Echo, Discard, or Character Generator protocols to test and debug applications	Configuring the MISC Server Chapter 33. on page 1091		MISCSERV

Security Considerations

The following sections contain information about security considerations, including migration information for TN3270 SSL Client Authentication support.

IP Security (IPSEC) Considerations

IPSEC is used to provide authentication, integrity, and data privacy for data transferred over an IP network. IP Security in SecureWay Communications Server allows the OS/390 to be used as part of a Virtual Private Network (VPN). A VPN is an extension of an enterprise's private intranet across a public network, such as the internet, to create a secure tunnel.

IPSEC is enabled in SecureWay Communications Server for OS/390 by specifying the FIREWALL option on IPCONFIG in the TCPIP.PROFILE. See "Chapter 4. Configuring the TCP/IP Address Space" on page 97 for more information about the IPCONFIG statement.

IPSEC is configured by using the OS/390 Firewall Technologies product. OS/390 Firewall Technologies can be used to configure an IPSEC policy that is used to govern what data can be sent or received by CS for OS/390 as well as if the data is authenticated and/or encrypted. For more information on OS/390 Firewall Technologies and how to configure IPSEC in CS for OS/390, refer to *OS/390 Firewall Technologies Guide and Reference*.

Consider the following points when IPSEC is enabled in CS for OS/390:

- You should specify DATAGRAMFWD on the IPCONFIG statement when CS for OS/390 is acting as a firewall between two networks.
- If you configure VIPA addresses in your IPSEC policy, you must also specify SOURCEVIPA on the IPCONFIG statement.
- Because SOURCEVIPA does not apply to dynamic VIPA addresses, you cannot use dynamic VIPA addresses in your IPSEC policy.
- When you configure a mixture of secure and non-secure adapters for CS for OS/390 and filter rules in your IPSEC policy do not have an interface value of BOTH, you should ensure that all routes to the destinations in a single filter rule go through adapters with the same security level (for example, either secure or non-secure). Refer to *OS/390 Firewall Technologies Guide and Reference* for more information about filter rules and configuring security attributes of adapters.
- Path MTU discovery is disabled for all packets using an IPSEC tunnel.

Security Product Considerations

A sample JCL job is provided in the hlq.SEZAINST member, EZARACF, to perform the Security Product resource definitions and define OMVS segments required for

the TCP/IP address and major servers. See Wave 2A Customizations in the Program Directory or ServPac Installing Your Order for more information about customizing and executing EZARACF.

Migration Considerations

TN3270 provides the capability to use SSL client authentication.

Current TN3270 processing continues to work unchanged.

If you want to exploit the client authentication support provided in CS for OS/390 V2R8, you must ensure the following:

- Your TN3270 clients support Client Authentication.
- Your TN3270 clients have obtained client certificates from a certificate authority that is defined in your server's keyring data base.
- Your TCP/IP profile has been updated to specify the CLIENTAUTH parameter.

If you want SAF certificate verification support (CLIENTAUTH SAFCERT), you must also do the following:

- Ensure that installed SAF product supports client certificates.
- Define RACDCERT as an authorized TSO command in the IKJTSOxx member.
- Ensure that the DIGTCERT class is active.
- Ensure that your client certificates have been defined to your security product and marked as *TRUSTED*.
- Consider RACLISTing the DIGTCERT class (for best performance).
- If allowing users to self register their certificates with RACF®, users need (at the minimum) READ authority to IRR.DIGTCERT.ADD in the FACILITY class.

If you plan to provide port-specific profiles in your security product to specify the users that can access a particular TN3270 port, do the following:

- Ensure your security product has the SERVAUTH class defined (RACF provides this support in V2R8).
- Add the profile and access list information to your security product and ensure the profile name follows the following format:

  ```
  EZB.TN3270.sysname.tcpname.PORTnnnnn
  ```

  where nnnnn is the port number with leading zeros specified.

  Note: The minimum access level that users added to the access list must be granted is READ access.
- Ensure the SERVAUTH class is active.
- Refresh the SERVAUTH RACLIST after changes to the profiles.

TN3270 provides the capability to use SSL Client authentication. To RACLIST THE DIGTCERT class, issue SETROPTS RACLIST(DIGTCERT) command. If the DIGTCERT class is RACLISTed, remember to refresh the RACLIST after adding/changing certificates (SETROPTS RACLIST(DIGTCERT) REFRESH). If planning to use the wildcard function in the port related profile name, also make sure generic profile support has been enabled for the SERVAUTH class. Installations with HTTPS for OS390 installed should consider using the self register web page support. Information on self registering client certificates with RACF using HTTPS for OS390 can be found in SYS1.SAMPLIB(IRR31933) (section 10) and SYS1.SAMPL IB(RACINSTL).

For more information on RACF related topics, refer to *Security Server (RACF) Administrator's Guide.*

Chapter 2. Customization and Administration Overview

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

The following section provides a high-level overview of the procedures for customizing and administering CS for OS/390. It includes information on:

- Updating your procedure library (PROCLIB)
- Starting and stopping the servers and the TCP/IP address space
- Controlling TCPIP and the Telnet server interactively with VARY TCPIP
- Running multiple instances (or copies) of TCP/IP

General Customization Procedure

Steps to Install and Customize CS for OS/390 along with other elements of OS/390:

1. Install CS for OS/390
2. Customize CS for OS/390
3. Update the VTAM® application definitions
4. Test and verify your configuration
5. Accept the product configuration

If you plan to use OS/390 UNIX sockets, see *OS/390 UNIX System Services Planning* for more information about setting up your system.

Step 1: Install CS for OS/390

Before you begin the installation:

- Read *OS/390 Planning for Installation* to help you plan the installation and migration of CS for OS/390.
- Be sure you understand the data set naming conventions used in TCP/IP. You can find this information in “Configuration Data Sets and HFS Files” on page 13.
- Consult the *OS/390 Program Directory* (Considerations for Wave 2A Customization) for current information about the material, procedures, and storage estimates of the MVS image. The Program Directory also contains information about customization of certain SYS1.PARMLIB members, which must be completed before the initial program load (IPL) of the MVS image.

Install CS for OS/390 V2R8 with other elements of OS/390. If you use the ServerPac method of installation, see *OS/390 Installing Your Order*; if you use the CBPDO method of installation, see the *OS/390 Program Directory*. When appropriate, those two books will direct you back to this book to customize the TCP/IP data sets and procedures and verify their configuration.

Verifying the Initial Installation

Both the *OS/390 Program Directory* and *OS/390 Installing Your Order* contain step-by-step instructions that can be used to set up and verify a basic TCP/IP configuration with only the loopback address and a few key servers. For more

information regarding these instructions, refer to the information about Wave 2A Customizations in the *OS/390 Program Directory* or the chapter on verifying your installation in *OS/390 Installing Your Order*.

Step 2: Customize CS for OS/390

To customize TCP/IP you need to update the cataloged procedures and configuration data sets for the TCPIP address space, its clients, and servers.

CS for OS/390 runs as a started task in its own address space. Each of the servers runs in its own address space and is started with its own procedure. The TCPIP address space requires:

- A cataloged procedure in a system or recognized PROCLIB.
- A data set that provides configuration definitions for the TCPIP address space and includes statements affecting many of the servers. This data set is referred to as PROFILE.TCPIP.
- A data set to provide the parameters that are common across all clients. This data set is referred to as TCPIP.DATA.

Many of the servers also require other data sets for their specific functions.

Making SYS1.PARMLIB Changes

You need to make certain changes to SYS1.PARMLIB. These changes depend on which of the following installation methods you use:

ServerPac method
: After the file system is restored (through the RESTFS job), you will see that SeverPac has changed some of the PARMLIB members. These changes are listed in the "Jobs or Procedures that Have Been Completed for You" in the Product Information appendix in *ServerPac Installing Your Order*. Follow the instructions in that book to change the BPXPRMxx member of PARMLIB.

CBPDO method
: Change the PARMLIB members according to the instructions listed in the chapters that describe installation instructions for Wave2. The chapter contains two tables that describe changes to PARMLIB and changes to BPXPRMxx member.

Completion of these steps ensures that the applications and resources on the target system will function correctly at the new level.

The subsequent chapters in this book show you how to:

- Configure the TCPIP address space by updating the samples provided in *hlq*.SEZAINST(SAMPPROF) and *hlq*.SEZAINST(TCPIPROC)
- Configure the universal client parameters provided in *hlq*.SEZAINST(TCPDATA)
- Configure the site table, defined in *hlq*.HOSTS.LOCAL, to identify the internet names and addresses of your TCP/IP host
- Customize the TCP/IP Component Trace parameters by updating the CTRACE parameter in the PARM= field of the EXEC JCL statement in the TCP/IP started procedure. See "Specifying TCP/IP Address Space Parameters" on page 99 for instructions.

  (You can find a description of the MVS Component Trace support in the *OS/390 SecureWay Communications Server: IP Diagnosis*.)

- Specify the ENVAR parameter on the PARMS=CTRACE(CTIEZB00) keyword to override the resolver file. For more information on setting the environment variable RESOLVER_CONFIG using the ENVAR parameter, see “Considerations for Multiple Instances of TCP/IP” on page 57.
- Configure each of the servers you want to run. This might require:
  - Modifying sample procedures and adding them in your PROCLIB
  - Modifying the configuration data set, PROFILE.TCPIP
  - Adding port numbers to *hlq*.ETC.SERVICES
  - Modifying other data sets containing server-specific parameters

Use the checklist provided in Table 3 on page 29 to decide which servers to configure and which samples they require. You can find the sample procedures and data sets in *hlq*.SEZAINST or the HFS. Table 4 on page 44 and Table 5 on page 53 provide additional reference information you can use as you configure and customize each server.

You can find general information about starting, stopping, and dynamically controlling the servers in “Administration Overview” on page 261. Specific information about operating and administering each server is also provided in the chapter for that server.

Step 3: Configuring VMCF and TNF

You can configure Virtual Machine Communication Facility (VMCF) and TNF in two different ways: As restartable subsystems or as non-restartable subsystems.

Restartable Subsystems

Configuring VMCF and TNF as restartable subsystems has the following advantages:

- Error detection is provided when the subsystems do not seem to be initializing properly.
- You can change the system name on the restart.
- Commands are available to remove users from internal tables, display current users and to terminate the subsystem.

In summary, a restartable VMCF and TNF configuration provides better availability and is therefore recommended.

Note: The Pascal socket interface makes use of the IUCV/VMCF services for a limited set of inter-address space communication flows. As a result, if you are using any applications (provided by IBM or others) that use the Pascal socket API, you must insure that the VMCF and TNF subsystems are active before the applications are started. TCP/IP provides several applications and commands that exploit these interfaces, such as the SMTP and LPD servers and the PING, TRACERTE, REXEC, RSH, and remote printing commands.

If you choose to use restartable VMCF and TNF, follow these steps:

1. Update your IEFSSN*xx* member in SYS1.PARMLIB with the TNF and VMCF subsystem statements required by TCP/IP:

   ```
   TNF
   VMCF
   ```

2. Update your SCHED*xx* member in SYS1.PARMLIB with the following entries:

   ```
   PPT PGMNAME(MVPTNF) KEY(0) NOCANCEL NOSWAP PRIV SYST
   ```

```
PPT PGMNAME(MVPXVMCF) KEY(0) NOCANCEL NOSWAP PRIV SYST
```

3. Add procedure EZAZSSI to your system PROCLIB. A sample of this procedure is located in the data set *hlq*.SEZAINST (where *hlq* is the high-level qualifier for the TCP/IP product data sets in your installation).

   ```
   //EZAZSSI PROC P=''
   //STARTVT EXEC PGM=EZAZSSI,PARM=&P
   ```

4. Start VMCF and TNF using the procedure EZAZSSI before starting TCP/IP during an MVS IPL as follows:

   ```
   S EZAZSSI,P=nodename
   ```

   Replace *nodename* with the NJE nodename of your MVS system.

Non-Restartable Subsystems

If you will not be using restartable VMCF and TNF, you should update your IEFSSN*xx* member in SYS1.PARMLIB with the following subsystem cards required by TCP/IP:

```
TNF,MVPTSSI
VMCF,MVPXSSI,nodename
```

Do not use the sample IEFSSN as is, since the comments are not valid in PARMLIB. A modified form of the last two lines must be placed in the IEFSSN*xx* PARMLIB member. Replace nodename on the VMCF line with the NJE node name of your MVS system.

VMCF Commands

The VMCF commands let you display the names of the current users of VMCF and TNF, and if necessary, remove names from the name lists.

Note: Removing names from the name lists and stopping either subsystem can have unpredictable results, if done hastily. Use the REMOVE and stop (P) commands carefully and only as a last resort.

If you remove a user, the application is not cancelled, nor is the connection severed. In other words, the ″removed″ application may remain active in the system, and may subsequently Abend 0D6/0D4/0C4, or cause TCP/IP to hang. A user that is removed from VMCF may still be a user of TNF and even TCP/IP, and vice versa.

To terminate users and stop VMCF or TNF properly, follow the steps below:

1. Display the current users of the subsystems, using one of the following:

   ```
   F VMCF,DISPLAY,NAME=*
   F TNF,DISPLAY,NAME=*
   ```

2. Terminate those users. If termination fails, use the REMOVE command as a last resort to force them from the name list.
3. Stop the subsystem, using one of the following commands:

   ```
   P VMCF
   P TNF
   ```

   If the P command fails, use one of the following commands:

   ```
   FORCE ARM VMCF
   FORCE ARM TNF
   ```

Following are descriptions of the commands:

F TNF,DISPLAY,NAME=[name|*]
Displays the named user (or all (*) users) of TNF, sorted by asid

F TNF,REMOVE,NAME=[name|*]
Removes either the named user (or all (*) users) from the TNF internal tables

P TNF Requests TNF to terminate

F VMCF,DISPLAY,NAME=[name|*]
Displays the named user (or all (*) users) of VMCF, sorted by name

F VMCF,REMOVE,NAME=[name|*])
Removes either the named user (or all (*) users) from the VMCF internal tables

P VMCF
Requests VMCF to terminate

Following are sample commands:

```
F TNF,DISPLAY,NAME=TCPV3
F VMCF,DISPLAY,NAME=*
F TNF,REMOVE,NAME=FTPSERV
F VMCF,REMOVE,NAME=*
P TNF
```

Common VMCF Problems

Following are some common VMCF problems:

- VMCF or TNF fail to initialize with an 0C4 abend.

  This is probably an installation problem; check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.
- Abends 0D5 and 0D6 after REMOVEing a user.

  This is probably because the application is still running and using VMCF. We do not recommend that users be removed from VMCF or TNF without first terminating the affected user.
- VMCF or TNF do not respond to commands.

  This is probably because one or both of the non-restartable versions of VMCF or TNF are still active. To get them to respond to commands, stop all VMCF/TNF users, FORCE ARM VMCF and TNF, then use EZAZSSI to restart.
- VMCF or TNF cannot be stopped.

  This is probably because users still exist in the VMCF and TNF lists. Use the *F VMCF,DISPLAY,NAME=** and *F TNF,DISPLAY,NAME=** commands to identify those users who are still active. Then either cancel those users or remove them from the lists using the *F VMCF,REMOVE* and *F TNF,REMOVE* commands.

IUCV/VMCF Considerations

The IUCV/VMCF inter-address space communication API enables applications running in the same MVS image to communicate with each other without requiring the services of the TCP/IP protocol stack. The VMCF/TNF subsystems provide these services which are still available in CS for OS/390. Several components of TCP/IP in CS for OS/390 continue to make some use of these services for the purpose of inter-address space communications. These include:

- The AF_IUCV domain sockets for the TCP/IP C socket interface The AF_IUCV domain enables applications executing in the same OS/390 image and using the

TCP/IP C socket interface to communicate with each other using a socket API, but without requiring the services of the TCP/IP protocol stack, as no network flows result in these communications. This is quite different from the more common AF_INET domain that enables socket communication over a TCP/IP network. AF_IUCV sockets continue to be supported in CS for OS/390.

An example of a TCP/IP-provided application that exploits AF_IUCV sockets is the SNMP Query Engine component (SQESERVE). The OS/390 UNIX socket library provides a similar functionality to the AF_IUCV domain sockets with its AF_UNIX domain. Users creating new applications should consider using AF_UNIX domain sockets.

- The Pascal socket interface also makes use of the IUCV/VMCF services for a limited set of inter-address space communication flows. As a result, any applications (provided by IBM or others) that use the Pascal socket API also still have a requirement for the VMCF/TNF subsystems. TCP/IP provides several applications and commands that exploit these interfaces, such as the SMTP, LPD, and the TELNET, PING, TRACERTE, HOMETEST, TESTSITE, RSH, REXEC, and LPR commands.

Therefore, in CS for OS/390 you must continue to configure and start the VMCF and TNF subsystems as you did in TCP/IP V3R2. However, because the VMCF/TNF subsystems are no longer used to directly communicate with the TCP/IP protocol stack in CS for OS/390, the amount of CPU they will consume will be significantly lower than in the TCP/IP V3R2 environment.

Step 4: Update the VTAM Application Definitions

The following TCP/IP applications use VTAM:

- SNALINK
- SNALINK LU6.2
- Telnet
- X.25 NPSI Server

You must update the VTAM definitions for Telnet and any other of these applications that you configure on your system. You can find example VTAM definitions for each of these applications in their respective chapters.

hlq.SEZAINST(VTAMLST) contains a sample of the VTAM definitions for Telnet applications. You should copy this member, update it, and add it to the ATCCON*xx* member of VTAMLST. This will ensure that the Telnet applications are activated when VTAM is started.

Because the TCP/IP LU code cannot handle multiple concurrent sessions, you must code SESSLIM=YES for each Telnet LU defined to VTAM. Otherwise, if SESSLIM=NO, menu or session manager applications that use 'return session' processing might cause session termination.

Step 5: Test and Verify Your Configuration

To verify that your configuration is correct, start the TCPIP address space.

1. Start the TCPIP address space
2. Verify your host name and address configuration
3. Optionally, verify your X Window System installation.

Start the TCPIP Address Space

Enter the MVS START command from the operator's console to start TCP/IP, specifying the member name of your cataloged procedure. This will start the TCPIP address space and any of the servers you have defined in the AUTOLOG statement in PROFILE.TCPIP. For example, if the procedure to start the TCPIP address space was called TCP1 in your PROCLIB, you would enter:

```
START TCP1
```

Verify Host Name and Address Configuration

Verify your host name and address configuration with HOMETEST. This program will test the system configuration defined by the HOSTNAME, DOMAINORIGIN, and NSINTERADDR statements in the TCPIP.DATA data set. In addition, it will check the IP addresses specified in the HOSTNAME statement in TCPIP.DATA against the HOME list in PROFILE.TCPIP, issue a warning message if any HOSTNAME addresses are missing from the HOME list, and report the result of the search order for TCPIP.DATA and FTP.DATA.

Note: The TSO PING command uses the Pascal socket API, so VMCF must be started for the command to be successful. If VMCF is not started, an ABEND0D6 can occur.

You can run the HOMETEST program in one of two ways:

- Run EZAVER in the *hlq*.SEZAINST data set. This job, shown in the following sample, executes HOMETEST as a batch job.

```
//EZAVER JOB (accounting,information),programmer.name,
//            MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A
//*
//*
//*   TCP/IP for MVS
//*
//*   5655-HAL (C) Copyright IBM Corp. 1992, 1994
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by GSA ADP
//*   Schedule
//*   Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//*   Function: host table configuration IVP
//*
//*
//VERIFY   EXEC  PGM=HOMETEST,REGION=800K
//SYSIN    DD    DUMMY
//SYSPRINT DD    SYSOUT=*
//*
//*   SYSTCPD can be used to over-ride the default search
//*   sequence for TCPIP.DATA.
//*
//*SYSTCPD DD    DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
//*
```

- Enter HOMETEST as a TSO command.

The HOMETEST command has no parameters.
If you have renamed your TCPIP.DATA data set to a name unique to your

installation then you must do one of the following to allocate that data set prior to issuing the HOMETEST command.

- Add a SYSTCPD DD statement to your TSO LOGON procedure
- Use the TSO ALLOCATE command

Sample HOMETEST Output: Following is a sample of the output from EZAVER. HOMETEST run interactively produces similar output. In this sample configuration, TCPIP.DATA and FTP.DATA were created as members of SYS1.TCPPARMS.

```
EZA0619I RUNNING IBM MVS TCP/IP CS for OS/390 V2R8 TCP/IP CONFIGURATION TESTER

EZA0620I THE TCP/IP SYSTEM PARAMETER FILE USED WILL BE "SYS1.TCPPARMS(TCPDATA)".
EZA0621I THE FTP CONFIGURATION PARAMETER FILE USED WILL BE "SYS1.TCPPARMS(FTPDATA)".

EZA0602I TCP HOST NAME IS: MVSA.TCP.RALEIGH.IBM.COM

EZA0605I USING HOST TABLES TO RESOLVE MVSA.TCP.RALEIGH.IBM.COM
EZA0611I THE FOLLOWING IP ADDRESSES CORRESPOND TO TCP HOST NAME:
         MVSA.TCP.RALEIGH.IBM.COM
EZA0612I 9.67.116.12
EZA0612I 9.67.116.38

EZA0614I THE FOLLOWING IP ADDRESSES ARE THE HOME IP ADDRESSES DEFINED
         IN PROFILE.TCPIP:
EZA0615I 9.67.116.12
EZA0615I 9.67.116.38

EZA0618I ALL IP ADDRESSES FOR MVSA.TCP.RALEIGH.IBM.COM ARE IN THE HOME LIST

EZA0622I HOMETEST WAS SUCCESSFUL - ALL TESTS PASSED
```

Verify the X Window System Installation

Support is provided for two versions of the X Window System and the corresponding OSF/Motif. The current support, provided as part of the base IP support in CS for OS/390, is for X Window System Version 11 Release 6 and OSF/Motif Version 1.2. Support for X Window System Version 11 Release 4 and OSF/Motif Version 1.1 is available as feature HTCP34X.

Verify the X Window X11R4 System Installation: X Window X11R4 System is installed with the other target libraries. The macro or headers go into the target library data set *hlq*.SEZACMAC. To verify the installation of the X Window System:

1. Specify your workstation IP address by adding a record (such as the following) to your XWINDOWS.DISPLAY data set.

   ```
   royal.csc.ibm.com:0.0
   ```

   In this example, `royal.csc.ibm.com:0.0` is the name of the host running the X Window System server.

   Note: No leading blanks are allowed in this record.
2. On the workstation running the X Window System server, issue an XHOST command specifying the name of your MVS system.
3. Run the program with the XSAMP1 command.

Verify the X Window X11R6 System Installation: To verify the installation of the X Window X11R6 System:

1. Ensure that a host (the workstation) with an X Window System server that supports X11R6 is properly configured and reachable by the MVS system. From the workstation, use Telnet to access the MVS system, and open an OS/390 UNIX shell on the MVS system.

2. From the OS/390 UNIX shell, export the DISPLAY environment variable using either the network name or the qualified IP address of the workstation as shown in the following example:

   ```
   export DISPLAY=royal.csc.ibm.com:0.0
   ```

   In this example, *royal.csc.ibm.com* is the name of the workstation running the X Window server. *:0.0* indicates the display, and is specified this way in almost all cases.
3. Authorize the MVS system to access the workstation by executing the XHOST command, and specify either the name of the MVS system or '+' as shown in the following example.

   ```
   xhost +
   ```

   Note: The *+* option will turn off security for this workstation and allow any X client to display here.
4. The sample X Window clients are shipped in the directory */usr/lpp/tcpip/X11R6/Xamples/demos*. Change into this directory. There are four sample program directories, xsamp1, xsamp2, xsamp3, and pexsamp. Change to the xsamp1 directory. Verify that there are two files named Makefile and xsamp1.c, and then execute the following command:

   ```
   make
   ```
5. Execute the program using the following command:

   ```
   xsamp1
   ```
6. The OS/390 UNIX shell should block as another window is opened. Verify the workstation is displaying a new window. The xsamp1 client will display a blank window for 60 seconds and then exit, taking its window with it. The OS/390 UNIX shell should no longer be blocked.

Step 6: Accept the Product Installation

Instructions for accepting the product installation are in the Program Directory. After you have verified your TCP/IP configuration, return to the Program Directory and follow the instructions provided there.

Cataloged Procedures and Configuration Data Sets

Table 4 on page 44 lists the cataloged procedures used by the TCP/IP functions and shows how each procedure gets the parameters it needs. These parameters can be passed directly in the procedure or they can come from configuration data sets and other data sets in the system.

The names of some of these data sets can be explicitly allocated using the JCL statements shown. The names of other data sets are hard-coded in the TCP/IP programs and, except for the high-level qualifier, cannot be changed. Table 5 on page 53 provides additional information about the configuration data sets.

Updating Your Procedure Library

Depending on your TCP/IP configuration and MVS system, you might not need to install all of the procedures shown in Table 4 on page 44. Additional information about each procedure is provided in the chapter for that server or function.

As you configure the TCPIP address space and each server, examine the sample procedures, copy them into your system PROCLIB or a recognized PROCLIB, and modify them to suit your installation.

Naming Your Procedure Members

Be aware that the name on the PORT statement in *hlq*.PROFILE.TCPIP must match the member name of the cataloged procedure you use to start that server or address space.

If you copy the sample procedure to a member with the same name as the one appearing on the PROC statement, the names on the AUTOLOG and PORT statements in SAMPPROF will not need modification.

Specifying Job Step Wait Time

The sample JCL procedures use TIME=1440 as a parameter on the EXEC statement to override Job Step Wait Timing. This is done to avoid unwanted system 522 abends.

Cataloged Procedures

Table 4. TCP/IP Cataloged Procedures

Function	Cataloged Procedures
Domain Name Server	**Procedure name:** NAMED **Sample:** SEZAINST(NAMED) **Purpose:** Starts the Domain Name Server. **Parameters:** Passed in the following configuration data set: *hlq***.NSMAIN.DATA** Samples: SEZAINST(EZADNSCP) or (EZADNSCC) Required, can be dynamically or explicitly allocated. Recommend you explicitly allocate with //SYSDNSD DD. **SEZAINST(NSMAINSC)** Required for caching only name servers. Data set name passed on CACHINGONLY statement in NSMAIN.DATA. *userid***.MASTER.DATA** Required for authoritative name servers.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
FTP	**Procedure name:** EZAFTPAP **Sample:** FTPD **Purpose:** Starts the FTP server. **Parameters:** Passed in the following configuration data set: *hlq*.**FTP.DATA** Sample: SEZAINST(FTCDATA)—client; SEZAINST(FTPSDATA)—server Optional, can be dynamically or explicitly allocated. Recommend you explicitly allocate with //SYSFTPD DD.
Kerberos	**Procedure name:** MVSKERB, ADM@SERV **Sample:** SEZAINST(MVSKERB), SEZAINST(ADM@SERV) **Purpose:** Starts the Kerberos authentication server. **Parameters:** Passed in the following configuration data sets: *hlq*.**ETC.SERVICES** Sample: SEZAINST(SERVICES). Required, can be explicitly allocated with //SERVICES DD. *userid*.**KRB.CONF** Sample: SEZAINST(KRBCONF). Required, cannot be explicitly allocated. **ADM@SRV.ADM@ACL.ADD** Sample: SEZAINST(ADMADD). Required, cannot be explicitly allocated. **ADM@SRV.ADM@ACL.GET** Sample: SEZAINST(ADMGET). Required, cannot be explicitly allocated. **ADM@SRV.ADM@ACL.MOD** Sample: SEZAINST(ADMMOD). Required, cannot be explicitly allocated.
MAKESITE	**Command name:** No sample procedure provided. **Purpose:** Runs MAKESITE to create your site tables (*userid*.HOSTS.INFO and *userid*.HOSTS.ADDRINFO data sets). **Parameters:** Whether run as a batch job or interactively, parameters are passed in the following data set: *hlq*.**HOSTS.LOCAL** Sample: SEZAINST(HOSTS). HLQ for input and output can be passed as a parameter.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
MISC Server	**Procedure name:** MISCSERV **Sample:** SEZAINST(MISCSERV) **Purpose:** Starts the server that supports the Echo, Discard, and Character Generator protocols. **Parameters:** Passed in the startup procedure.
NCPROUTE	**Procedure name:** NCPROUT **Sample:** SEZAINST(NCPROUT) **Purpose:** Starts the NCPROUTE server to use NCP as a dynamic RIP router. **Parameters:** Passed in the following configuration data sets: **NCPROUTE profile** Sample: SEZAINST(EZBNRPRF). Optional, cannot be dynamically allocated. Must be explicitly allocated with //NCPRPROF DD. **NCPROUTE gateways** Sample: SEZAINST(EZBNRGW). Optional, cannot be dynamically allocated. Must be defined with GATEWAYS_PDS in EZBNRPRF. *hlq*.**ETC.SERVICES** Sample: SEZAINST(SERVICES). Required, can be explicitly allocated with //SERVICES DD.
NCS	**Procedure names:** NRGLBD, LLBD **Samples:** SEZAINST(NRGLBD), SEZAINST(LLBD) **Purpose:** Start the 2 Network Computing System servers. **Parameters:** Passed in the following data sets: **STANDARD.TCPXLBN** No sample provided. Dynamically allocated. **ADM@SRV.LLB@LL.DATABASE** Created automatically by the LLBD procedure. Required, cannot be explicitly allocated. **ADM@SRV.LLB@LG.DATABASE** Created automatically by the NRGLBD procedure. Required, cannot be explicitly allocated.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
NDB	**Procedure names:** PORTS, PORTC **Samples:** SEZAINST(PORTSPRC), SEZAINST(PORTCPRC) **Purpose:** Start the NDB Port Manager, start the NDB port client and NDB server(s). **Parameters:** Does not use any configuration datasets. All parameters are passed on the EXEC statements of the cataloged procedures.
OMPROUTE	**Procedure name:** OMPROUTE **Sample:** SEZAINST(EZAORCFG) **Purpose:** Starts OMPROUTE to handle dynamic routing with OSPF or RIP. **Parameters:** Passed in the following configuration data sets: **OMPROUTE profile** Required. Searched for in the first file found below: • OMPROUTE_FILE environment variable • /etc/omproute.conf • *hlq*.ETC.OMPROUTE.CONF

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
OROUTED	**Procedure name:** OROUTED **Sample:** SEZAINST(OROUTED) **Purpose:** Starts the OROUTED server to handle dynamic routing with RIP gateways. **Parameters:** Passed in the following configuration data sets: **OROUTED profile** Sample: SEZAINST(EZARTPRF). Optional. First file found according to the following search order: 1. ROUTED_PROFILE environment variable 2. /etc/routed.profile 3. *hlq*.ROUTED.PROFILE **OROUTED gateways** Sample: SEZAINST(EZARTGW). Optional. First file found according to the following search order: 1. GATEWAYS_FILE environment variable 2. /etc/gateways 3. *hlq*.ETC.GATEWAYS *hlq***.ETC.SERVICES** Sample: SEZAINST(SERVICES). Required. Searched for in the first file found below: 1. /etc/services 2. userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES
OS/390 UNIX Portmapper	**Procedure name:** PORTMAP **Sample:** SEZAINST(OPORTPRC) **Purpose:** Starts the OS/390 UNIX Portmapper server that makes RPC programs available across the network through TCP/IP. **Parameters:** Passed in the following configuration data sets: *hlq***.ETC.RPC** Sample: SEZAINST(ETCRPC). Required, cannot be explicitly allocated.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
OS/390 UNIX Remote Execution Server	**Procedure name:** RXSERVE **Sample:** SEZAINST(RXPROC) **Purpose:** Starts the server that executes the OS/390 UNIX Remote Execution Command Daemon. **Parameters:** Does not use any configuration data sets. All parameters are passed on the EXEC statement of RXPROC.
Remote Print Server	**Procedure name:** LPSERVE **Sample:** SEZAINST(LPSPROC) **Purpose:** Starts the server that executes the Line Print Daemon (LPD). **Parameters:** Passed in the following configuration data set: *hlq*.**LPD.CONFIG** Sample: LPDDATA. Required, can be dynamically allocated or explicitly allocated using a parameter on the PROC statement in the cataloged procedure.
Service Policy Agent	**Procedure name:** PAGENT **Sample:** SEZAINST(PAGTPROC) **Purpose:** Starts the Policy Agent to handle service policy definitions. **Parameters:** Passed in the following configuration data sets: **PAGENT profile** Sample: SEZAINST(PAGENT). Optional. First file found according to the following search order: 1. -c startup option 2. PAGENT_CONFIG_FILE environment variable 3. /etc/pagent.conf 4. hlq.PAGENT.CONF

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
RSVP	**Procedure name:** RSVP **Sample:** SEZAINST(RSVPD) **Purpose:** Starts the RSVP Agent to provide resource reservation services. **Parameters:** Passed in the following configuration data sets: **PAGENT profile** Sample: SEZAINST(RSVPDCF). Optional. First file found according to the following search order: 1. -c startup option 2. RSVPD_CONFIG_FILE environment variable 3. /etc/rsvpd.conf 4. hlq.RSVPD.CONF
SLA Subagent	**Procedure name:** PAGTSNMP **Sample:** SEZAINST(PAGTSNMP) **Purpose:** Starts the SLA subagent to provide service policy performance monitoring. **Parameters:** Does not use any configuration data sets. All parameters are passed as startup options.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
SMTP	**Procedure name:** SMTP **Sample:** SEZAINST(SMTPPROC) **Purpose:** Starts the server that executes the Simple Mail Transfer Protocol. **Parameters:** Passed in the following configuration data sets: **SMTP configuration** Sample: SEZAINST(SMTPCONF). Required, must be explicitly allocated with //CONFIG DD. **SMTP security table** No sample provided. Optional, must be explicitly allocated with //SECTABLE DD. **SMTP rules** No sample provided. Optional, must be explicitly allocated with //SMPTRULE DD. **SMTPNJE.HOSTINFO** No sample provided. Optional, must be explicitly allocated with //SMPTNJE DD. *mailfiledsprefix*.**SECURITY.MEMO** No sample provided. Required, cannot be explicitly allocated.
SNALINK LU0	**Procedure name:** SNALINK **Sample:** SEZAINST(SNALPROC) **Purpose:** Initializes the SNALINK LU0 interface. **Parameters:** Does not use any configuration data sets. All parameters are passed on the EXEC statement of SNALPROC.
SNALINK LU6.2	**Procedure name:** TCPIPL62 **Sample:** SEZAINST(LU62PROC) **Purpose:** Initializes the SNALINK LU6.2 interface. **Parameters:** Passed in the following data set: **LU6.2 configuration** Sample: SEZAINST(LU62CFG). Required, must explicitly allocate with //LU62CFG DD. In this procedure, you must explicitly allocate TCPIP.DATA with //SYSTCPD DD.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
SNMP Server	**Procedure names:** OSNMPD **Samples:** SEZAINST(OSNMPDPR) **Purpose:** Start the Simple Network Management Protocol agent. **Parameters:** Passed in the following configuration data sets: **PW.SRC** No sample provided. Required. **SNMPTRAP.DEST** No sample provided. Optional **OSNMPD.DATA** Optional. Sample: /usr/lpp/tcpip/samples/osnmpd.data **SNMPD.CONF** Required if SNMPv3 protocols are used. Sample: /usr/lpp/tcpip/samples/snmpd.conf For the search order for the above data sets, see “Chapter 22. Configuring Simple Network Management Protocol (SNMP)” on page 823. The SNMP agent uses some statements from TCPIP.DATA. The SNMP subagent uses some statements in the PROFILE.TCPIP.
TCPIP	**Procedure name:** TCPIP **Sample:** SEZAINST(TCPIPROC) **Purpose:** Starts the TCPIP address space. **Parameters:** Passed in the following configuration data sets: **PROFILE.TCPIP** Sample: SEZAINST(SAMPPROF). Required, can be dynamically or explicitly allocated. Recommend you explicitly allocate with //PROFILE DD. **TCPIP.DATA** Sample: SEZAINST(TCPDATA). Required, can be dynamically or explicitly allocated. For more information about TCPIP.DATA search order, see “Chapter 1. Before You Begin” on page 7.

Table 4. TCP/IP Cataloged Procedures (continued)

Function	Cataloged Procedures
X.25 NPSI Server	**Procedure name:** TCPIPX25 **Sample:** SEZAINST(X25PROC) **Purpose:** Initilalizes the X.25 NPSI interface. **Parameters:** Passed in the following data set: **X.25 server configuration** Sample: SEZAINST(X25CONF). Required, must explicitly allocate with //X25IPI DD. This procedure also uses other NPSI configuration data sets which are specified through VTAM: **SVC configuration** Sample: SEZAINST(X25VSVC). Required. **NPSIGATE** Sample: SEZAINST(NPSIGATE). Optional. **NPSIDATE** Sample: SEZAINST(NPSIDATE). Optional. In this procedure, you must explicitly allocate TCPIP.DATA with //SYSTCPD DD.

Configuration Data Sets

The following table lists the configuration data sets used by the TCP/IP servers and functions. It includes the name of the sample and the usage of the data set.

Table 5. TCP/IP Configuration Data Sets

Data Set/Search Order	Copied From	Usage
ADM@ACL.ADD	SEZAINST(ADMADD)	Used by the Kerberos server, authorizes remote administrators to add database entries.
ADM@ACL.GET	SEZAINST(ADMGET)	Used by the Kerberos server, authorizes remote administrators to query database entries.
ADM@ACL.MOD	SEZAINST(ADMMOD)	Used by the Kerberos server, authorizes remote administrators to modify database entries.
DNSDB	SEZAINST(DNSDB)	Sets up DB2® storage group for the Domain Name Server.
DNSTABLE	SEZAINST(DNSTABLE)	Defines the primary and secondary zones for the DB2 tables used by the Domain Name Server.
DNSTCP	SEZAINST(DNSTCP)	Sets up DB2 SQL tables for the Domain Name Server.

Table 5. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
ETC.PROTO 1. /etc/protocol 2. userid.ETC.PROTO 3. *hlq*.ETC.PROTO	usr/lpp/tcpip/samples/protocol	Used to map types of protocol to integer values to determine the availability of the specified protocol. Required by several CS for OS/390 components.
ETC.RPC	SEZAINST(ETCRPC)	Defines RPC applications to the Portmapper function.
ETC.SERVICES 1. /etc/services 2. userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES	usr/lpp/tcpip/samples/services	Establishes port numbers for servers using TCP and UDP. Required for OS/390 UNIX SNMP, OROUTED, and OMPROUTE (if the RIP protocol is used).
FTP.DATA 1. //SYSFTPD 2. *userid/jobname*.FTP.DATA 3. /etc/ftp.data 4. SYS1.TCPPARMS(FTPDATA) 5. *hlq*.FTP.DATA	SEZAINST(FTCDATA) for the client and (FTPSDATA) for the server	Overrides default FTP client and server parameters for the FTP server.
HOSTS.LOCAL (or /etc/hosts)	SEZAINST(HOSTS)	Contains host names and IP addresses, used for non-DNS name resolution.
KRBCONF	SEZAINST(KRBCONF)	Identifies the hosts running the Kerberos authentication server.
LPD.CONFIG	SEZAINST(LPDDATA)	Configures the Line Printer Daemon for the Remote Print Server.
LU62CFG	SEZAINST(LU62CFG)	Provides configuration parameters for the SNALINK LU6.2 interface.
MASTER.DATA	No sample provided	DNS database input required for authoritative name servers.
NPSIDATE	SEZAINST(NPSIDATE)	Operates the TCP/IP X.25 NCP Packet Switching Interface.
NPSIFC	SEZAINST(NPSIFC)	Supports GATE MCH Fast Connect option for X.25 NCP Packet Switching Interface.
NPSIGATE	SEZAINST(NPSIGATE)	Supports GATE MCHs for X.25 NCP Packet Switching Interface.
NSMAIN.DATA	SEZAINST(EZADNSCP) SEZAINST(EZADNSCP)	Name server initialization parameters, includes primary and secondary database names and cache pool sizes.
NSMAINSC	SEZAINST(NSMAINC)	Authoritative DNS locations for caching only name servers.
PAGENT.CONF	SEZAINST(PAGENT)	Controls and regulates service differentiation.

Table 5. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
PROFILE.TCPIP 1. //PROFILE 2. *job_name.node_name*.TCPIP 3. *hlq.node_name*.TCPIP 4. *job_name*.PROFILE.TCPIP 5. *hlq*.PROFILE.TCPIP	SEZAINST(SAMPPROF)	Provides TCP/IP initialization parameters and specifications for network interfaces and routing.
RSVPD	SEZAINST(RSVPDCF)	Allows for end-to-end resource reservation on behalf of applications.
SMTPCONF	SEZAINST(SMTPCONF)	Provides configuration parameters for the Simple Mail Transfer Protocol.
SMTPNOTE	SEZAINST(SMTPNOTE)	Defines note parameters for Simple Mail Transfer Protocol.
TCPIP.DATA	SEZAINST(TCPDATA)	Provides parameters for TCP/IP client programs. **Note:** The search order depends on the type of application (OS/390 UNIX or native MVS) - for details, see "Search Order and Configuration Files for the TCP/IP Stack" on page 16.
TNDBCSCN	SEZAINST(TNDBCSCN)	Provides configuration parameters for Telnet 3270 Transform support.
VTAMLST	SEZAINST(VTAMLST)	Defines VTAM applications and their characteristics. Entries required for Telnet, SNALINK LU0, SNALINK LU6.2, and X.25 NPSI Server.
X25CONF	SEZAINST(X25CONF)	Provides configuration parameters for the X.25 NCP Packet Switching Interface.
X25VSVC	SEZAINST(X25VSVC)	Provides switched virtual circuit configuration for the X.25 NCP Packet Switching Interface.

Customizing TCP/IP Messages

The messages for every TCP/IP server program are compiled and linked with the program and reside in an internal message repository. Some of the server programs that are written in the C language also have their messages in external data sets. You can edit these external message data sets to translate the messages to another language or customize them to suit your installation.

How to Access the Message Data Sets

The procedures for these servers have a special DD statements that point to the external message data set. If you are going to override the internal messages and use external customized messages, you need to remove the comment from the appropriate DD statement and ensure it points to the correct data set.

The following table shows the servers that have external messages, the DD statement used, and the name of the message data set delivered with the system:

Server	DD Statement	Data Set
NCPROUTE	//MESSAGE	SEZAINST(EZBNRMSG)
SNMP	//MSSNMPMS	SEZAINST(MSSNMP)
MISC Server	//MSMISCSR	SEZAINST(MSMISCSR)
DNS	//MSNSMAIN	SEZAINST(MSNSMAIN)

Message Format

This information describes the message format and message text.

Message Text

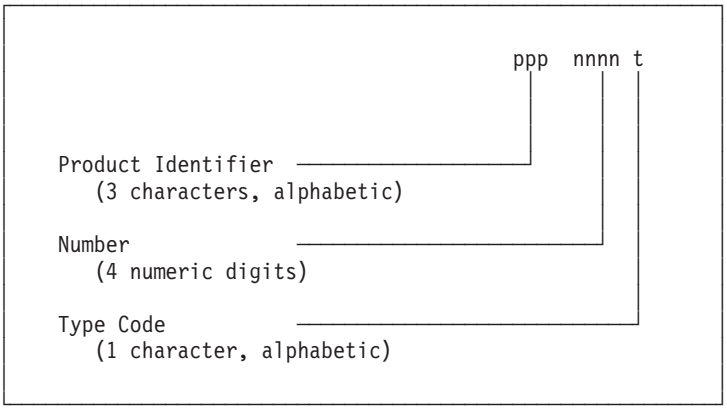
The message text might include special characters for the variable fields that are converted when the message is printed or displayed and control characters that affect the message format. The conversion characters start with a percent sign (%) and the control characters start with a backslash (\). These are all standard notations for the C language print function. The messages might also contain comments which start with `/*` and end with `*/`

In the following simulated message, the control character `\n` forces a new line to print and the string variables, represented by `%s`, are converted in the order they are passed from the program.

```
29999  I  Command %s received from user %s\n
```

Message Format

The following diagram explains the syntax for TCP/IP message IDs on the host:



The ***product identifiers*** (ppp) for TCP/IP are EZA, EZB, and EZY. The ***number*** (nnnn) indicates a unique 4-digit numeric value assigned to the message by product. The ***type*** (t) indicates the severity assigned to the message.

Rules for Customizing the Messages

The general rule for customizing or translating messages is to **only** change the text portion of the message.

- Do not change the MARGIN, PRODUCT and COMPONENT definitions at the top of the data set. These are required definitions for the program. For example, these entries at the top of the MISC server message data set should not be changed:

```
MARGINS(1,72)
PRODUCT EZA
COMPONENT MSC
```

- Do not change the message numbers and the severity code. These parts of the message have specific meaning; if you change them the program may not work correctly.
- Do not change the conversion characters. These indicate that the program is passing data, the type of data it is passing. and the appropriate way to display or print this data. For example, do not change or delete %s and %d in the following message:

  ```
  4858 W   "Route from %s in unsupported address family %d\n"
  ```

- You can reorder the variables that are passed in the message. For example, you can reverse the order of the two string variables that are passed when translating a message by specifying the new order of the arguments in parentheses following the message text:

  Before: `29999I Command %s received from user %s\n`

  After: `29999I Utilisador %s envio instrucion %s\n (2, 1)`

  The result would be `EZY9999I Utilisador MANNY envio instrucion FTP` instead of `EZY9999I Command FTP received from user MANNY.`

- Watch for any program parameters or keywords that might be in the message text. In most cases, you should not translate them.

  For example, in the following message, `'active'` is a keyword used in the gateway definition and should not be translated:

  ```
  4851 E   "First two elements must be 'active' for active gateway\n"
  ```

Considerations for Multiple Instances of TCP/IP

Prior to CS for OS/390 V2R5, several reasons existed for running multiple TCP/IP stacks within a single image. Two of these reasons are:

- Increased capacity and availability

  To fully take advantage of a multiple-processor system, multiple TCP/IP stacks would be configured to take advantage of the multiple processors.

  The SecureWay Communications Server for OS/390 TCP/IP stack is a multi-processor capable stack, which means that it can concurrently exploit all available processors on a system. Starting multiple stacks will not yield a significant increase in throughput.

- Isolation of OS/390 UNIX applications from TCP/IP applications

  Some installations would configure two TCP/IP stacks, one defined under the OS/390 UNIX environment and one that is not.

  In SecureWay Communications Server for OS/390, a single TCP/IP stack supports both environments, OS/390 UNIX and TCP/IP. The OS/390 UNIX environment is now an integral part of the TCP/IP stack. As a result, the TCP/IP stack can no longer be configured to not support the OS/390 UNIX environment.

In addition, running multiple SecureWay Communications Server for OS/390 TCP/IP stacks requires additional system resources, such as storage, CPU cycles, and DASD. It also adds a significant level of complexity to the system administration tasks for TCP/IP. Therefore, running multiple SecureWay Communications Server for OS/390 TCP/IP stacks in the same MVS image is not recommended.

For help in configuring multiple instances of TCP/IP, refer to the OS/390 UNIX manuals. In particular, read the chapter on “Setting Up for Common INET Sockets”

in *OS/390 UNIX System Services Planning*. The information provided in this section is specific to running multiple instances of SecureWay Communications Server for OS/390.

Common INET Physical File System (C-INET PFS)

The following are examples of Physical File Systems (PFS) in an OS/390 UNIX environment:

- Network (AF_INET) sockets PFS
- Common INET sockets PFS
- Hierarchical File System (HFS)

The Network (AF_INET) PFS and the Common INET PFS handle socket requests from C programs and OS/390 UNIX applications. The HFS PFS lets applications access files, then passes file requests from an OS/390 UNIX application, through DFSMS/MVS®, to the HFS where traditional files or special character files are located.

Depending on the number of stacks you want to run on the sockets interfaces, you can use the Network (AF_INET) or the Common INET. The Network (AF_INET) supports one TCP/IP stack at a time. It is used when applications communicate through a single stack. Common INET is used when applications communicate through multiple stacks.

You can specify your choice of Network (AF_INET) or Common INET on the NETWORK DOMAINNAME and FILESYSTYPE statements of SYS1.PARMLIB(BPXPRMxx). For more information about the BPXPRMxx statements, refer to *OS/390 UNIX System Services Planning*. Examples of coding BPXPRMxx statements are provided in *OS/390 Program Directory* in the chapter that describes installation instructions for Wave 2, and in the ServerPac book, *OS/390 Installing Your Order*.

Port Management Overview

When there is a single transport provider, and the relationship of server to transport provider is 1:1, port management is relatively simple. Using the PORT statement, the port number can be reserved for the server in the PROFILE.TCPIP for that single transport provider.

Port management becomes more complex in an environment where there are multiple transport providers (multiple instances of CS for OS/390) and a potential for multiple combinations of the same server (for example, OS/390 UNIX and tn3270/tn3270E Telnet).

In a multiple transport provider environment, the following questions need to be resolved:

- Is the server generic or does the server have an affinity for one instance of the transport providers?
- How can ports be reserved across multiple transport providers? When is the port reservation determined by MVS rather than by the jobname, procedure name, or userid?
- How can you synchronize between BPXPARMS and PORTRANGE for ephemeral port reservation?

- How can CS for OS/390 distinguish between two different instances of Telnet (OS/390 UNIX TELNET and tn3270/tn3270E Telnet)?

Generic Server Versus Server with Affinity for a Specific Transport Provider

The following sections describe the differences between generic servers and servers with affinities for specific transport providers.

Generic Server: A generic server, a server without an affinity for a specific transport provider, provides service to any client on the network. (See Figure 2.) FTP is an example of a generic server. The transport provider is merely a connection linking client and server. The service File Transfer is not related to the internal functioning of the transport provider, and the server can communicate concurrently over any number of transport providers.

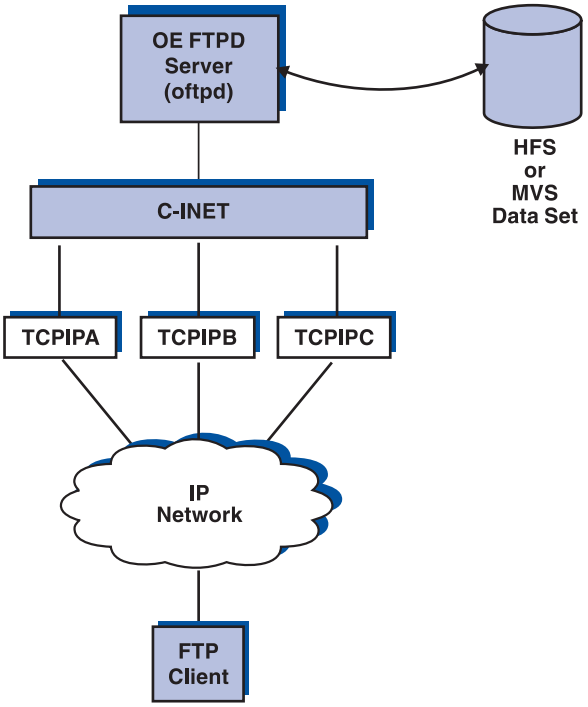


Figure 2. Generic Server

Server with an Affinity for a Specific Transport Provider: When the service is related to the internal functioning of the transport provider (for example, TELNET, OROUTED, OSNMPD, and onetstat, a command), there must be an explicit binding of the server application to the chosen transport provider. (See Figure 3 on page 60.) There must also be a way to specify the single transport to be chosen.

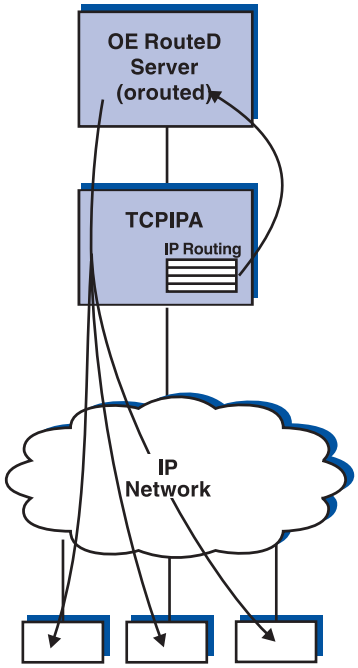


Figure 3. Server with Affinity for a Specific Transport Provider

With the exception of applications that use the socket API provided by CS for OS/390, other IBM-supplied applications that use the OS/390 UNIX socket API and that must bind to a specific transport provider use the new OS/390 UNIX socket call `setibmopt()` (see the *OS/390 C/C++ Run-Time Library Reference*) to specify which TCP they have chosen. A new C function `__iptcpn()`, described in the *OS/390 C/C++ Run-Time Library Reference*, enables the application to search the TCPIP.DATA file to find the name of the specific TCP/IP. (See Figure 4.)

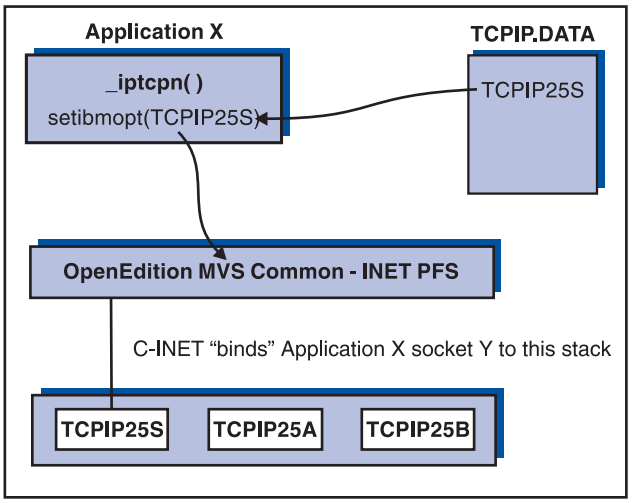


Figure 4. Example of Binding an Application to a Specific Transport Provider

Port Number Conflicts in a C-INET Environment

In CS for OS/390, you can configure multiple TCP/IP stacks in a single MVS image using the C-INET feature. In a C-INET configuration, an application using the OS/390 UNIX socket interface can get transparent access to all the TCP/IP protocol

stacks configured under C-INET. For example, when an application coded to OS/390 UNIX sockets performs a SOCKET/BIND/LISTEN in a C-INET environment, the request is propagated by C-INET to all the TCP/IP stacks. This application can then service client requests that arrive into any of the configured TCP/IP stacks without having any awareness of this fact. This type of application is often referred to as a ″generic server/daemon″.

The following servers/daemons shipped by CS for OS/390 V2R8 are generic servers/daemons:

- FTPD
- OS/390 UNIX RSHD
- OS/390 UNIX REXECD
- OS/390 UNIX TELNETD
- TFTPD
- DHCP
- TIMED
- OS/390 UNIX Portmap

OS/390 UNIX RSHD, REXECD and TELNETD are usually started by the INETD daemon which is shipped as part of the OS/390 UNIX. Since INETD is also a generic daemon, any server processes started by INETD inherently become generic servers as well.

Although generic servers can help reduce complexity in some C-INET environments, there are also cases where their use can present some undesirable features. Consider a configuration where the use of both the OS/390 UNIX REXEC and the MVS REXEC servers is required on the same MVS image. This could be accomplished by designating an alternate port specification for one of these two servers. For example, the MVS REXEC server could continue to use the well known port of 512, and the OS/390 UNIX REXEC server could use 1512 as its port. This configuration would allow both servers to use a single TCP/IP stack. REXEC clients, typically on other TCP/IP hosts, would then need to specify the port value that matched the environment they wanted to reach (OS/390 UNIX or TCP/IP). This works fairly well if the platform where the REXEC client command is issued provides an easy way to specify a port other than the well known port. On some platforms this can only be accomplished by modifying the ETC/SERVICES file. This can sometimes be a cumbersome process, especially if that TCP/IP host needs to access both the OS/390 UNIX and MVS REXEC servers or REXEC servers on other platforms.

An alternative solution to this problem is to try to have the well known port be used by both the OS/390 UNIX and MVS REXEC servers on the same MVS image. Obviously, this is not possible on a single TCP/IP stack. A solution to this problem could be to configure two TCP/IP stacks on a single MVS image: one of the stacks would be designated as the OS/390 UNIX server stack and the other as the TCP/IP server stack. Note that in OS/390 V2R8, it is not possible to designate a TCP/IP stack as not being enabled for OS/390 UNIX. Consider further that since a C-INET environment is also required to execute multiple TCP/IP stacks, generic servers are typically serviced by all stacks available. This scenario requires the ability to bind the MVS REXEC server to one stack and the OS/390 UNIX REXEC server to the other. The MVS REXEC server always has affinity to the TCP/IP stack specified in the TCPIPJOBNAME parameter on its TCPIP.DATA file so this is not a problem. However, as discussed earlier, the OS/390 UNIX REXECD is started by way of

INETD which is a generic server. Therefore, in this scenario, we need to be able to have INETD, a generic daemon, have affinity to a specific stack.

This affinity can be accomplished by use of the **_BPXK_SETIBMOPT_TRANSPORT** environment variable.

This environment variable, when set, has a similar effect to the **setibmopt()** function call provided by C/C++ compiler and described in the OS/390 V2R4.0 C/C++ Run-Time Library Reference. This variable can be set in the JCL for a started procedure or batch job that executes an OS/390 UNIX C/C++ program to indicate which TCP/IP stack instance the application should bind to. TCP/IP applications that require affinity to a specific TCP/IP stack, like OSNMPD and OROUTED, use the **setibmopt()** function call directly. The **_BPXK_SETIBMOPT_TRANSPORT** environment variable basically provides the ability to bind a generic server type of application to a specific stack.

For example, if you had two TCP/IP stacks configured under C-INET, one named TCPIP and the other TCPIPOE, and you wanted to start an FTPD server instance that was associated with TCPIPOE you could modify the FTPD procedure as follows:

```
//FTPD   PROC MODULE='FTPD',PARMS='TRACE'
//FTPD   EXEC PGM=&MODULE,REGION=7M,TIME=NOLIMIT,
//       PARM=('POSIX(ON) ALL31(ON)',
//      'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE")',
//      '/&PARMS')
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*
//*           /etc/ftp.data
//*           SYSFTPD DD statement
//*           jobname.FTP.DATA
//*           SYS1.TCPPARMS(FTPDATA)
//*           tcpip.FTP.DATA
//*
//*       If no FTP.DATA file is found, FTP default values are used.
//*       For information on FTP defaults, see the Customization
//*       and Administration Guide and TCP/IP OE MVS Applications
//*       Feature Guide.
//*SYSFTPD DD DISP=SHR,DSN=TCPIP.SEZAINST(FTPSDATA)
//*
//*      SYSTCPD explicitly identifies which file is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the JCL of
//*      the server.  The file can be any sequential data set,
//*      member of a partitioned data set (PDS), or HFS file.
//SYSTCPD DD DISP=SHR,DSN=SYS1.TCPPARMS(TCPDATA)
//*
//*      SYSFTSX explicitly identifies which file is to be used
//*      for the EBCDIC-ASCII translation table.  The file can
//*      be any sequential data set, member of a partitioned data
//*      set (PDS), or HFS file.
//SYSFTSX DD DISP=SHR,DSN=TCPV34.STANDARD.TCPXLBIN
```

All the parameters specified prior to '/' in the parm statement are processed by the C/C++ run time library. Parameters to be passed to the FTPD program must appear

after '/'. Also note how the parameters were split over three lines in this example since they could not fit on a single line. Another example follows with JCL for the started procedure for INETD:

```
//INETD  PROC
//*************************************************************
//INETD  EXEC PGM=BPXBATCH,
//*      PARM='PGM /usr/sbin/inetd -d /etc/inetd.conf'
//       PARM='PGM /usr/sbin/inetd    //''USER1.INETD.CONF'''
//*
//STDERR  DD PATH='/tmp/inetd.debug.stderr',
//  PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//  PATHMODE=SIRWXU
//STDOUT  DD PATH='/tmp/inetd.debug.stdout',
//  PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//  PATHMODE=SIRWXU
//STDENV   DD  DISP=SHR,DSN=USER1.INETD.ENVIRON
```

The STDENV dataset would contain the _BPX_SETIBMOPT_TRANSPORT variable as shown below:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE
```

In the previous example, INETD was also passed its configuration file as a parameter. In our example, this file is an MVS data set rather than an HFS file, therefore it requires the additional '//' and quotes that the example shows.

Multiple instances of INETD are not allowed even if each instance is bound to a different TCP/IP stack. This is an INETD restriction, not a TCP/IP related one. Therefore, if you decide to make INETD have affinity to a specific stack then that is the only INETD instance that you will be able to have running in that MVS image.

Notes:

1. The **_BPXK_SETIBMOPT_TRANSPORT** variable should only be specified for a generic server type of application.

   If specified for a non-generic server and/or non-OS/390 UNIX application it will not have any effect.
2. The name specified for **_BPXK_SETIBMOPT_TRANSPORT** must match the jobname associated with the TCP/IP stack.

   If the name specified does not match the jobname of any TCP/IP stacks defined for C-INET the application will receive an OS/390 UNIX return code of X'3F3' and a return value of X'005A' and may be accompanied by the following message:

   ```
   EDC8011I A name of a PFS was specified that either is not configured or is not a Sockets PFS.
   ```

   If the name specified does not match the jobname of any currently active TCP/IP stack defined under C-INET the application will receive an OS/390 UNIX return code of X'70' and a return value of X'0296' and may be accompanied by the following message:

   ```
   EDC5112I Resource temporarily unavailable.
   ```

Port Reservation across Multiple Transport Providers

When there are multiple transport providers, be sure to synchronize the PORT statements in each of the PROFILE.TCPIP files to ensure that the port reservations for each stack match the port definitions for the servers that will be using that stack.

Ephemeral Ports: When running with multiple transport providers, just as it is necessary to synchronize PORT reservations for specific applications across all stacks, it is required to synchronize reservations for port numbers that will be dynamically assigned across all stacks. These are the ephemeral ports above 1024, which are assigned by the stack when none is specified on the application `bind()`. To reserve a group of ports in the PROFILE.TCPIP, use PORTRANGE. Specify the same PORTRANGE for every stack. In addition, you need to let the OS/390 UNIX C-INET know which ports are guaranteed to be available on every stack. Use BPXPARMS to do this.

- PROFILE.TCPIP
  - PORTRANGE 4000 1000 TCP OMVS ; Reserved for OMVS
  - PORTRANGE 4000 1000 UDP OMVS ; Reserved for OMVS
- BPXPRMxx parmlib member
  - NETWORK DOMAINNAME(AF_INET)
  - ...
  - INADDRANYPORT(4000)
  - INADDRANYCOUNT(1000)

SMF Accounting Issues

Many installations rely on the MVS component System Monitoring Facility (SMF) for job accounting and for performance analysis. TCP/IP can create SMF118-type SMF records for certain events. If you are running multiple stacks, SMF does not always allow you to distinguish among them. Consider the following issues:

- There is no stack identity in SMF118 records. SMF records that are written by the system address space or by standard servers may be identified as belonging to one stack or another, based on address space naming conventions.
- SMF records written by client address spaces cannot be identified as belonging to a single stack via this method.
- The only technique currently available to distinguish among records written by various client address spaces is to assign unique SMF118 record subtype intervals to each stack:
  - **FTP Server** One or nine subtypes in FTP.DATA
  - **Telnet Server** Two subtypes on TELNETPARMS
  - **API** Two subtypes on SMFPARMS
  - **FTP, Telnet Client** One subtype on SMFPARMS

If you choose to assign subtypes, there will be an obvious impact on your local accounting programs. SMF118 subtype changes and additions must be coordinated with persons responsible for managing the use of SMF.

Selecting a Stack

Socket application programs in a multi-stack environment must contend with the following:

- How does the socket program select which TCP/IP stack to use for its socket communication?
- How does the TCP/IP resolver code executing in the socket application address space decide which TCP/IP resolver configuration data sets to allocate?

In order to answer these questions, we need to distinguish between standard servers and clients (those that come with the CS for OS/390 product), and other socket application programs, including those you might have written yourself.

Standard Servers and Clients

The anchor configuration data set is TCPIP.DATA data set. This is the main resolver configuration data set with information on host name, domain origin, and so on. It holds the TCPIPJOBNAME parameter, which identifies the TCP/IP stack to use, and the DATASETPREFIX parameter, which is used by the resolver code and other services when allocating configuration data sets (HOSTS.SITEINFO, HOSTS.ADDRINFO, ETC.SERVICES, ETC.PROTO, and STANDARD.TCPXLBIN). See “Appendix G. Protocol Number and Port Assignments” on page 1193 for more information about ETC.PROTO and ETC.SERVICES.

The key to selecting both a specific stack and resolver configuration data sets is to control which TCPIP.DATA data set a standard server or client address space allocates.

Standard servers and clients search for TCPIP.DATA in the following sequence:

1. //SYSTCPD DD (the SYSTCPD DD-name)
2. *jobname*.TCPIP.DATA for batch jobs and started task, or *userID*.TCPIP.DATA, for TSO users
3. SYS1.TCPPARMS(TCPDATA)
4. TCPIP.TCPIP.DATA

OS/390 UNIX servers and clients will search for TCPIP.DATA in the following sequence:

1. ENVIRONMENT VARIABLE ″RESOLVER_CONFIG=file/dataset″
2. /etc/resolv.conf
3. //SYSTCPD DD

   This is not valid for ″fork-ed″ processes.

   Note: The SYSTCPD DD data set is not propagated from the parent process over the fork() or exec() function calls.
4. *userid*.TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
5. SYS1.TCPPARMS(TCPDATA)
6. TCPIP.TCPIP.DATA

Non-Standard Servers and Clients

Non-standard servers and clients also use TCPIP.DATA to decide which resolver configuration data sets to allocate, but they may or may not use the TCPIPJOBNAME parameter to select a stack. This choice depends on the following factors:

- The socket API used to create the program
- The release of TCP/IP for MVS they use

If you run sockets programs from other products or vendors, you may want to know which sockets API was used to develop the program, and which techniques, if any, the program uses to specify the name of the TCP/IP system address space. With CS for OS/390, this task is made easier than with previous releases. As long as

application programs that use an CS for OS/390 socket library do not specify anything specific on calls `setibmopt()` or `INITAPI`, the TCPIPJOBNAME from a TCPIP.DATA data set will be used as the last resort for finding a TCP/IP system address space name.

Table 6 depicts the differences that prevail in stack selection depending on the socket API and the version of TCP/IP for MVS under which you are running the socket program.

Table 6. How Your Own Socket Programs Select a Stack

C Sockets	Sockets Extended	Pascal Sockets	REXX Sockets
SETIBMOPT or TCPIPJOBNAME from TCPIP.DATA	TCPNAME on INITAPI or TCPIPJOBNAME from TCPIP.DATA	TCPIPJOBNAME from TCPIP.DATA	TCPIPJOBNAME from TCPIP.DATA
Sockets Extended programs might have a configuration option to specify the TCP/IP system address space name, or the might interrogate the available stacks via the getibmopt() call.			

Note that in TCP/IP Version 3 Release 1, a Sockets Extended program does not use TCPIP.DATA to select a stack; it must specify a value on the TCPNAME parameter on an INITAPI call.

In TCP/IP Version 3 Release 2 and subsequent releases, a Sockets Extended program does not have to call INITAPI. If INITAPI is not called, an implicit INITAPI is performed with the value taken from TCPIPJOBNAME in a TCPIP.DATA data set. If INITIAPI is called, a TCPNAME of space results in the TCPIPJOBNAME keyword value being used as the TCP/IP system address space name.

In an OS/390 UNIX environment, the TCP/IP system address space is not selected by the socket application program, but rather by the AF_INET PFS.

TCP/IP TSO Clients

TSO client functions can be directed against any of a number of TCP/IP stacks. Obviously the client function must be able to find the TCPIP.DATA appropriate to the stack of interest at any one time. Two methods are available for finding the relevant TCPIP.DATA:

- Add a SYSTCPD DD statement to your TSO logon procedure. The issue with this approach is that a separate TSO logon procedure per stack is required, and users have to log off TSO and log on again using another TSO logon procedure in order to switch from one stack to another.
- Use one common TSO logon procedure without a SYSTCPD DD statement. Before a TSO user starts any TCP/IP client programs, the user has to issue a TSO ALLOC command wherein the user allocates a TCPIP.DATA data set to DDname SYSTCPD. To switch from one stack to another, the user simply has to de-allocate the current SYSTCPD allocation and allocate another TCPIP.DATA data set.
- Combine the first and second methods. Use one logon procedure to specify a SYSTCPD DD for a default stack. To switch stacks, issue TSO ALLOC to allocate a new SYSTCPD. To switch back, issue TSO ALLOC again with the name that was on the SYSTCPD DD in the logon procedure. The disadvantage to this approach is that the name that was on the SYSTCPD DD is ″hidden″ in the logon procedure and needs to be retrieved or remembered.

The last method can be implemented by creating a small REXX program for every TCP/IP stack on your MVS system. For each stack create a REXX program with the name of the stack (for example, T18A or T18B). Whenever TSO users want to use the T18A stack, they run the T18A REXX program. Any TCP/IP functions invoked thereafter will use the T18A stack for socket communication. If users want to switch to the T18B stack, they run the T18B REXX program. See Figure 5 for an example.

```
/* REXX  "T18B" */
/*********************************************************************/
/*                                                                   */
/* Switch TSO Address Space to use the T18B Stack.                   */
/* Subsequent NETSTAT command will be directed toward                */
/*  the T18BTCP stack.                                               */
/*                                                                   */
/*********************************************************************/
Say 'Switching to T18BTCP stack'

msgstat = msg()
z = msg("OFF")
"FREE FI(SYSTCPD)"
"ALLOC FI(SYSTCPD) DA('TCPIP.T18B.TCPPARMS(TCPDATA)') SHR"
z = msg(msgstat)

exit(0)
```

Figure 5. REXX Program to Switch TSO User to Another TCP/IP Stack

Selecting Configuration Data Sets

The resolver code and other services that execute as part of the socket program address space to service calls such as `gethostbyname()`, `getservbyname()` and `getprotobyname()` allocates one or more configuration data sets to service these calls. All socket programs, including standard servers and clients and homegrown socket programs, need access to one or more of these configuration data sets. In addition to TCPIP.DATA, the configuration data sets are as follows. (See also Figure 6 on page 68.)

- **MVS**

  HOSTS.ADDRINFO and HOSTS.SITEINFO
- **OS/390 UNIX**

  /etc/resolv.conf and /etc/hosts

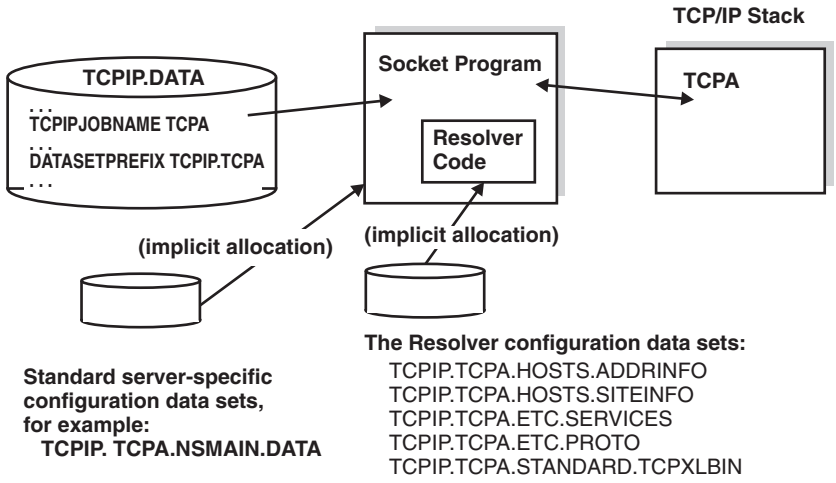


Figure 6. Selecting Configuration Data Sets

The resolver code uses the DATASETPREFIX from the selected TCPIP.DATA configuration data set to search for the resolver configuration data sets. In addition to allocating the resolver configuration data sets, TCP/IP standard servers may use the DATASETPREFIX value when they search for server-specific configuration data sets. For example, when the name server searches for an NSMAIN.DATA data set, it looks for DATASETPREFIX.NSMAIN.DATA in one of the search steps.

Sharing Resolver Configuration Data Sets

The general recommendation is to use separate DATASETPREFIX values for each stack and create separate copies of the required configuration data sets; at the very least, create separate copies of the resolver configuration data sets. For a test and a production stack, however, you would probably use different DATASETPREFIX values. However, if the stacks are functionally identical, you may share the same DATASETPREFIX values and many of the same configuration data sets. You need separate TCPIP.DATA data sets because of the two different TCPIPJOBNAMEs. On the other hand, you may choose to share the resolver configuration data sets between the stacks by using the same DATASETPREFIX value in each TCPIP.DATA data set. (See Figure 7 on page 69.)

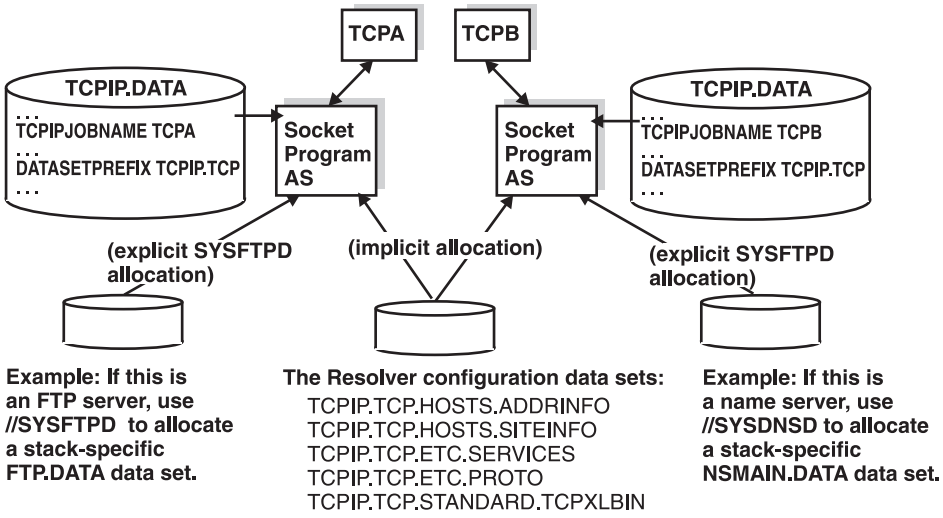


Figure 7. Sharing DATASETPREFIX

In addition to separate TCPIP.DATA data sets, separate /etc/resolv.conf files might also be necessary. If this is the case, use the environment variable RESOLVER_CONFIG to point to the appropriate resolver information.

Exercise caution if servers use DATASETPREFIX to allocate server-specific configuration data sets. Try to use explicit allocation as far as possible in your server JCL procedures. Most servers allow you to explicitly allocate their configuration data sets using DD statements.

Some servers may use DATASETPREFIX to create new data sets. Servers that do create new data sets allow you to specify an alternate data set prefix for the data sets that are created. NPF creates new sequential data sets with captured print data. NPF has a special keyword in NPF.DATA for this purpose; it is called NPFPRINTPREFIX. If this keyword is specified, NPF will use that as high level qualifier for newly created print data sets instead of taking the DATASETPREFIX value from TCPIP.DATA. Another example of a server that creates new data sets is the SMTP server.

Specifying BPXPRMxx Values for a C-INET Configuration

For a detailed description of parameters in SYS1.PARMLIB(BPXPRMxx), refer to *OS/390 UNIX System Services Planning* and *OS/390 MVS Initialization and Tuning Guide*.

```
/* AF_INET file system for sockets      */
/* Converged socket support - BPXTCINT  */

FILESYSTYPE TYPE(CINET)
            ENTRYPOINT(BPXTCINT)  1
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(10000)  2
        TYPE(CINET)
        INADDRANYPORT(10000)  3
        INADDRANYCOUNT(2000)
SUBFILESYSTYPE NAME(TCPIP1A)  4
               TYPE(CINET)
               ENTRYPOINT(EZBPFINI)  5
               DEFAULT  6
SUBFILESYSTYPE NAME(TCPIP1B)  4
               TYPE(CINET)
               ENTRYPOINT(EZBPFINI)
```

Figure 8. SYS1.PARMLIB(BPXPRMxx) for Converged Sockets

1 C-INET and BPXTCINT specify the use of converged sockets.

2 The number of MAXSOCKETS should be large enough to open new sockets for OS/390 UNIX applications.

3 INADDRANYPORT and INADDRANYCOUNT specify the first ephemeral port number and the range of ports for OS/390 UNIX. These values have to match the PORTRANGE definitions in your PROFILE data sets for both TCP/IP stacks.

4 A transport provider stack for converged sockets is specified with a SUBFILESYSTYPE statement. The NAME field must match the address space name for the TCP/IP started task as well as the TCPIPJOBNAME parameter in TCPIP.DATA. In our example, the name of the first stack is TCPIP1A and the name of the second stack is TCPIP1B.

5 EZBPFINI identifies a CS for OS/390 TCP/IP stack.

6 Keyword DEFAULT specifies which transport provider stack is to be used as the default stack for OS/390 UNIX. If DEFAULT is not specified, the first stack will be used as the default stack. The sequence of SUBFILESYSTYPE statements is arbitrary if one stack is identified with the keyword DEFAULT. TCPIP1A is the default stack in our example.

Chapter 3. Virtual IP Addressing

This chapter contains information about the following topics:

- Introduction to VIPA
- Moving VIPA (upon failure or TCP/IP)
- Static vs. Dynamic VIPAs
- Using static VIPAs
- Using Dynamic VIPAs
- Resolution of Dynamic VIPA conflicts
- Other considerations
- Dynamic VIPAs and routing protocols

Introduction to VIPA

Traditionally, an IP address is associated with each end of a physical link (or each point of access to a shared-medium LAN), and the IP addresses are unique across the entire visible network, which can be the Internet or a closed intranet. The majority of IP hosts have a single point of attachment to the network, but some hosts (particularly large server hosts) have more than one link into the network. A TCP/IP host which has multiple points of attachment also has multiple IP addresses, one for each link.

Within the IP routing network, failure of any intermediate link or adapter disrupts end-user service only if there is not an alternate path through the routing network. Routers can route IP traffic around failures of intermediate links in such a way that the failures are not visible to the end applications or IP hosts. However, an IP packet is routed based on ultimate destination IP address. If the adapter or link associated with a destination IP address fails, there is currently no way for the IP routing network to provide an alternate path to the destination TCP/IP and application. Client and server IP adapters and links thus constitute single points of failure. While this might be acceptable for a client host, where it is likely that only a single user will be cut off from service, a server IP link might serve hundreds or thousands of clients, all of whose services would be disrupted by a failure of the server link.

The Virtual IP Address (VIPA) addresses the problem of the single point of failure of an S/390 application host IP adapter or link by providing an IP address that is associated with an OS/390 TCP/IP stack without associating it with a specific physical network attachment. VIPA uses a virtual device and a virtual link. The virtual device will always be active and never see a failure. A VIPA address will be the home address for the virtual link, but there will be no physical network attachment associated with it.

To the routing network, a VIPA appears to be a host, a network, or a subnet that is reachable via the OS/390 TCP/IP through any of the physical interfaces of that TCP/IP. After an IP packet with a destination VIPA reaches the TCP/IP where the VIPA is defined, the TCP/IP recognizes that the packet belongs here and routes the packet up the stack to a higher layer protocol (for example, TCP or UDP). The key point is that the VIPA is an address of a TCP/IP that can be reached via any of the physical links of that TCP/IP. If one of the physical devices or links for that TCP/IP fails, IP traffic to or from that VIPA continues as long as there is an alternate path to

the TCP/IP via another physical interface. Static or dynamic routing protocols may be used to provide alternate paths. In general, OS/390 configured with VIPA provides the following functions:

- Automatic and transparent recovery from device failure.

  When a device (for example, 3172, or channel-attached 2216) fails, if there is another device that provides the alternate paths to the destination, and if other hosts are connecting to the VIPA addresses:

  – IP will detect the failure, find an alternate path for each network, and route 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 the active TCP connections that were using the failed device.

- Automatic and transparent recovery from adapter failure.

  Assume that multiple physical network adapters (for example, Token Ring or FDDI) are installed on a device, if there are multiple paths serving as alternate paths defined over these adapters to the destination, and if other hosts are connecting to the VIPA addresses:

  – IP will detect the failure, find an alternate path for each network, and route 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 the active TCP connections that were using the failed adapter.

- Recovery from OS/390 TCP/IP stack failure (where an alternate OS/390 TCP/IP stack has the necessary redundancy)

  Assume that an alternate OS/390 TCP/IP stack is installed to serve as a backup and VIPA is configured on the primary OS/390 TCP/IP stack. In case of a primary OS/390 TCP/IP stack failure, the backup OS/390 TCP/IP stack can be reconfigured to use the primary OS/390 TCP/IP stack's VIPA address. Client/server connections on the failed primary OS/390 TCP/IP stack will be disrupted but they can be reestablished on the backup OS/390 TCP/IP stack using the primary OS/390 TCP/IP stack's VIPA as the destination address. In addition, the temporarily reassigned VIPA address can be restored to the restored, primary OS/390 TCP/IP stack.

  Note: For requests or connections originating at an OS/390 TCP/IP stack, tolerance of device and adapter failures may be achieved by using the SOURCEVIPA option. With this option, VIPA addresses will be used as the source IP addresses in outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests.

Moving a VIPA (Upon Failure of TCP/IP)

While a VIPA provides non-disruptive rerouting of IP data for failure of a physical interface, failure of the TCP/IP or the associated OS/390 (including planned outages) will disrupt connections or UDP sessions to applications on that TCP/IP. While failure of the TCP connection or UDP session will be visible to the clients, the duration of the outage is determined by how long the client application is unable to reconnect to an appropriate server application. It is common in large enterprises to have multiple instances of such an application residing on different MVS images. Many clients will attempt to reconnect to the IP address that was in use at the time of the failure and do not return to DNS to resolve the name to an IP address for another server application. Therefore, if we could move the known IP address to a

functioning TCP/IP that supports an instance of the application, the clients would be able to reconnect and the perceived outage would be over.

IP addresses associated with a particular physical device are unavailable until the owning TCP/IP is restarted. However, a VIPA is not associated with any particular physical interface. If a failure of a TCP/IP is detected, and a suitable application already is active on another backup TCP/IP, the VIPA can be started on the other TCP/IP until the original TCP/IP can be restarted. Client/server connections on the failed TCP/IP stack will be disrupted, but they can be reestablished on the backup TCP/IP stack using the original TCP/IP stack's VIPA as the destination address. The activation of a VIPA on a backup stack is referred to as VIPA takeover.

Movement of a VIPA to a backup TCP/IP stack can be accomplished by using VARY OBEY on the other backup TCP/IP with a TCP/IP configuration file containing an appropriate set of DEVICE, LINK, and HOME statements, as well as appropriate BSDROUTINGPARMS statements or OMPROUTE configuration, depending on the routing daemon on the other TCP/IP.

If the TCP/IP configuration file with the statements defining the VIPA is created in advance, the transfer can be accomplished via appropriate automation, without requiring actual operator intervention. This procedure is documented in "Chapter 4. Configuring the TCP/IP Address Space" on page 97. It is up to the system programmer (for automation) or operator (for movement by operator intervention) to ensure that the VIPA is moved to a TCP/IP that has an appropriate server application.

In the absence of a failure, a VIPA is just like any other IP address, and routing for a VIPA is the same as for an IP address associated with a physical link. Such routing can be statically defined or dynamic. However, if a VIPA is to be moved, either new static definitions must be provided at the new TCP/IP, or dynamic routing protocols must be used.

As an alternative to moving a VIPA as described above, an OS/390 TCP/IP stack can be configured to enable dynamic movement of a VIPA upon stack failure to an appropriately configured backup stack within the same sysplex without operator intervention or customer programming for automation. See "Dynamic Movement of a VIPA (Dynamic VIPA Takeover Function)" on page 78.

Static vs. Dynamic VIPAs

OS/390 TCP/IP supports two kinds of VIPAs: static and dynamic. Static and Dynamic VIPAs may coexist on a given stack, or on different stacks within a Sysplex. Both static and Dynamic VIPAs eliminate dependence upon particular physical interfaces to an OS/390 stack, but there are differences in how they are configured and used.

Static VIPAs have the following charactistics:

- They can be allocated during TCP/IP initialization or VARY OBEY processing and are configured using DEVICE, LINK, and HOME statements along with BSDROUTINGPARMS statements or appropriate OMPROUTE configuration.
- Using the SOURCEVIPA configuration option, static VIPAs can be used as the source IP address for outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests. This provides tolerance of device and adapter failures for requests of connections originating at an OS/390 TCP/IP stack.

- Static VIPAs can be moved to a backup stack after the original owning TCP/IP stack has failed, by using VARY OBEY processing to configure the VIPA on the backup stack and updating the routers.
- The number or static VIPAs on a stack is limited only by the range of host IP addresses that can be used for VIPA and non-VIPA addresses.

See “Using Static VIPAs” for more detailed descriptions.

Dynamic VIPAs, introduced with OS/390 V2R8:

- Can be configured to be moved dynamically from a failing TCP/IP stack to a (properly configured) backup stack within the same sysplex without operator intervention or customer programming for automation
- Can be dynamically allocated by an application program running on a properly configured TCP/IP stack

A TCP/IP stack can have both dynamically moveable VIPAs and dynamically allocated VIPAs, but a given Dynamic VIPA will have one or the other capability, not both.

Unlike static VIPAs, Dynamic VIPAs:

- May not be used as a VIPA for SOURCEVIPA
- Are limited to 64 per stack
- Cannot be used with Enterprise Extender (see “Configuring Static VIPAs for an OS/390 TCP/IP Stack” for details)

See “Using Dynamic VIPAs” on page 77 for more detailed descriptions.

Using Static VIPAs

The following sections describe how to configure and use static VIPAs.

Configuring Static VIPAs for an OS/390 TCP/IP Stack

Assume that you want to configure a VIPA address in one OS/390 TCP/IP stack. Following are the configuration steps:

1. Add a VIPA device and link to the DEVICE and LINK statements. See “DEVICE and LINK Statement—Virtual Devices” on page 188.
2. Add a VIPA link to the HOME statement. See “HOME Statement” on page 207.
3. If the VIPA address is to be designated as a default local host, see “PRIMARYINTERFACE Statement” on page 235.

   Note: A VIPA link cannot be coded in the GATEWAY statement.
4. Directly-connected VIPA routes cannot be coded in the GATEWAY statement. For this reason, these routes cannot be displayed using the onetstat -r, NETSTAT ROUTE, onetstat -g, or NETSTAT GATE commands. However, if using a dynamic routing application (for example, OMPROUTE), the directly-connected VIPA routes can be viewed using the routing application’s display or modify command when displaying the contents of its internal routing table. Furthermore, indirectly-connected routes (either defined statically or learned dynamically) to VIPA addresses in other OS/390 TCP/IP stacks over non-VIPA links can be viewed using these onetstat or NETSTAT commands.

5. If tolerance of device and adapter failures is desired for requests or connections originating at an OS/390 TCP/IP stack, specify the SOURCEVIPA option in the IPCONFIG statement. For this option to work properly, the receiving nodes in the network must be configured to recognize the SOURCEVIPA addresses using the static or dynamic routing protocols. Otherwise, timeouts for the connection or request responses will occur as a result of the VIPA addresses not being network reachable. For more information on configuring SOURCEVIPA addresses, see “HOME Statement” on page 207.
6. For a VIPA host name resolution, replace the physical IP address with the VIPA address in the domain name servers.
7. Depending upon the routing protocols used, the routing paths using the physical network attachments to reach the destination hosts and the routing paths from the receiving nodes to reach the VIPA address as the destination are either defined statically or learned dynamically. For more information, see “Static VIPA and Routing Protocols” on page 77.

Notes:

1. Any physical links defined in an OS/390 TCP/IP stack must be defined in addition to the VIPA links to provide the physical network attachments.
2. If using static routing and multiple paths are defined in the network, the routers providing the paths must have the ability to switch to alternate routes in case of failures. Otherwise, connections will be disrupted.

Figure 9 on page 76 illustrates a sample configuration showing multiple network attachments using a single VIPA address. Depending upon the routing protocols used, the network or host routes in Router1 and Router2 are used to reach the destination VIPA address.

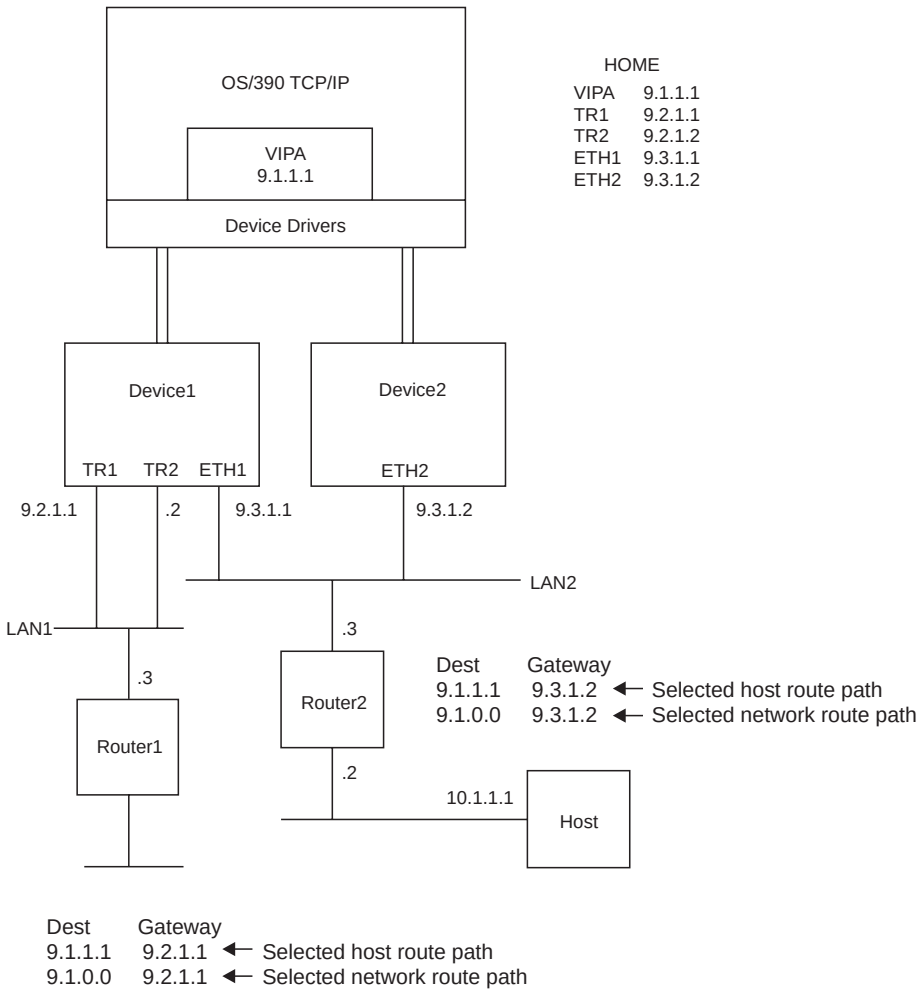


Figure 9. Single VIPA Configuration

Configuring Static VIPAs for Enterprise Extender

Defining a VIPA address is required if you are configuring a stack that will be used by VTAM for access to the IP network. You may specify multiple VIPA addresses. If multiple VIPA addresses are specified in the PROFILE.TCPIP data set, VTAM uses either the VIPA address specified on the VTAM IPADDR start option, or if not specified as a VTAM start option, the first VIPA address listed in the PROFILE.TCPIP data set. All other remote APPN nodes that wish to communicate with the VTAM that uses this stack for access to the IP network using Enterprise Extender must specify this VIPA address as the destination, either by explicitly coding this VIPA address or by coding this host's hostname if name-to-address resolution is available to the remote APPN node. Note that if remote APPN nodes use a hostname to define the destination of an Enterprise Extender connection, this VIPA address must be the only IP address returned by name-to-address resolution services for that hostname. For more information about Enterprise Extender, refer to *OS/390 SecureWay Communications Server: SNA Network Implementation Guide*

Planning for Static VIPA Takeover and Takeback

Because a VIPA is associated with an OS/390 TCP/IP stack and is not associated with a specific physical network attachment, it can be moved to a TCP/IP stack on the same OS/390 image or to another TCP/IP stack on another OS/390 image. Moving a static VIPA can be done manually by operator intervention or by customer-programmed automation. Movement of the static VIPA allows other hosts that were connected to the primary TCP/IP stack to reestablish connections with a backup TCP/IP stack using the primary TCP/IP stack's VIPA address. After the primary TCP/IP stack has been restored, the temporarily reassigned VIPA address can be restored.

Consider the following when backing up and restoring an OS/390 TCP/IP stack:

- All sessions between the clients and the server on the failing host will be disrupted.
- The client can use any ephemeral port number when re-establishing the connection to the backup server.
- Having a different port number for the backup and primary server is not recommended. If the backup server has a different port number than the original primary server (for example, port 101 rather than port 21 for FTP), the client must know to use 101 rather than 21. Different port numbers does work, but can cause administrative problems.

For more information on backing up and restoring an OS/390 TCP/IP stack, see the following topics depending upon the application used:

- If using static routing, see "GATEWAY Statement" on page 193.
- If using OROUTED, see "Configuring a Backup TCP/IP Stack with VIPA" on page 943 and "Restoring a Primary OS/390 TCP/IP Stack with VIPA" on page 944.
- If using OMPROUTE, see "Chapter 26. Configuring OMPROUTE" on page 949.

Static VIPA and Routing Protocols

Use the following guidelines for static VIPA routing protocols.

- If using static routes via the GATEWAY statement, ensure that there are multiple paths to reach a destination host. Likewise, ensure that there are multiple paths defined in a network that a host can use to reach the VIPA address as the destination. The multiple paths are used to route around device and adapter failures.
- If using OMPROUTE to manage routes dynamically using the OSPF/RIP protocols, see "Chapter 26. Configuring OMPROUTE" on page 949.
- If using OROUTED to manage routes dynamically using the RIP protocol, see "Chapter 25. Configuring the OROUTED Server" on page 917.

Using Dynamic VIPAs

Dynamic VIPA support allows:

- Dynamic movement of a VIPA from a failing TCP/IP stack to a backup stack
- Dynamic allocation of a VIPA by an application program

Caution

Dynamic VIPAs are IP addresses like all other IP addresses associated with a TCP/IP, and they appear as though they had been defined at the end of the HOME list. Because TCP/IP searches the HOME list for each incoming IP packet to determine whether the packet belongs to this stack, having too many IP addresses in the HOME list might adversely affect the performance of all incoming IP traffic. To minimize any performance impact, there is a limit of 64 Dynamic VIPAs at any one time on a single TCP/IP.

Configuring the Dynamic VIPA Support

Dynamic VIPA support can be configured at TCP/IP initialization or configured/modifed via VARY OBEY processing.

Unlike static VIPAs, Dynamic VIPAs are not configured using DEVICE, LINK, and HOME statements.

The configuration statements for the Dynamic VIPA support are contained within VIPADYNAMIC and ENDVIPADYNAMIC statements and consist of the following:

- VIPADEFINE and VIPABACKUP statements used to configure dynamically moveable VIPAs
- VIPARANGE used to specify a range of IP addresses which may be dynamically allocated as a VIPA by an application program
- VIPADELETE used to delete existing Dynamic VIPAs

The following sections discuss how these statements are used to provide the Dynamic VIPA support. For syntax details, see “VIPADYNAMIC Statement” on page 254.

Dynamic Movement of a VIPA (Dynamic VIPA Takeover Function)

Movement by network management automation or operator intervention is not always desirable. Operator intervention takes time and is subject to errors. Automation requires proper detection of the failure and is also prone to error if the failure does not result in the exact console message anticipated to indicate the failure.

In this release, the Dynamic VIPA Takeover function addresses this problem. It is important to understand that Dynamic VIPA Takeover does not introduce function that could not be accomplished by operator action or automation. However, Dynamic VIPA Takeover does not depend on human detection of the error or customer programming for automation. Dynamic VIPA Takeover is completely accomplished by the (properly configured) TCP/IP stacks.

Dynamic VIPA takeover is possible when a Dynamic VIPA is defined as active (via VIPADEFINE) on one stack and as backup (via VIPABACKUP) on another stack within the sysplex. When the stack on which the VIPA is active terminates, then the backup stack will automatically activate the VIPA and notify the routing daemon. If dynamic routing is not used for Dynamic VIPAs, the attached routing network might not be able to locate a Dynamic VIPA which has been moved dynamically to a new (backup) TCP/IP.

For dynamic takeover to be useful, the application(s) that service the VIPA address(es) must be available on the backup stacks. In the absence of the application, the VIPA will be active, but client connections will still fail.

Note: Because the same VIPA address cannot be active on more than one stack at a time, the applications that cannot bind to INADDR_ANY are not appropriate for dynamic takeover. However, applications that cannot bind to INADDR_ANY can still take advantage of the dynamic allocation of VIPAs by binding to a specific address or by using the EZBXFDVP utility program.

Planning for Dynamic VIPA Takeover

First, you should determine how the workload will be distributed among the backup stacks. If multiple applications are running on the active stack, the workload can be distributed among multiple stacks, or a single stack can receive all the application's workload. If a single stack will backup the active stack, a single VIPADEFINE can be used. If you want to distribute the workload among multiple stacks, use multiple VIPADEFINEs with corresponding VIPABACKUPs.

The following example shows how to implement a single stack backup for multiple applications.

Stack A:

```
Uses VIPADEFINE to define 9.67.113.105
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 9.67.113.105 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 9.67.113.105 as their destination address.
```

Stack B:

```
Uses VIPABACKUP to define 9.67.113.105 as backup for stack A.
Has a Web server running that binds to INADDR_ANY.
Has an FTP server running that binds to INADDR_ANY.
```

In the above scenario, when stack A goes down, stack B takes over all new connections for both the Web and FTP servers. FTP and Web client programs continue to use 9.67.113.105 as their destination address, but they now connect to stack B.

The following example shows how to implement a multiple stack backup for multiple applications.

Stack A:

```
Uses VIPADEFINE to define 9.67.113.105 and 9.67.100.100
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 9.67.113.105 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 9.67.100.100 as their destination address.
```

Stack B:

```
Uses VIPABACKUP to define 9.67.113.105 as backup for stack A.
Has a Web server running that binds to INADDR_ANY.
```

Stack C:

```
Uses VIPABACKUP to define 9.67.100.100 as backup for stack A.
Has an FTP server running that binds to INADDR_ANY.
```

In the above scenario, when stack A goes down, new connections for the Web server at 9.67.113.105 will connect with stack B, and new connections for the FTP server at 9.67.100.100 will connect with stack C.

Different Application Uses of IP Addresses and VIPAs

Not all IP-based server applications relate to IP addresses in the same way. Automated movement of VIPAs, and the planning for VIPA Takeover, must take this difference into account.

Some applications will accept client requests on any IP address (for example, they bind to INADDR_ANY such as TN3270 or Web servers). The distinguishing feature of such an application is the business function it provides (the particular set of SNA applications for TN3270 or the particular web pages for a Web server). If the business function is replicated across multiple OS/390 images in the Sysplex, as is often the case for distributed workload, the movement of the VIPA must merely be to one of the TCP/IPs supporting such a suitable application instance. This scenario is called the Multiple Application Instance Scenario. For the Multiple Application Instance Scenario, the TCP/IPs in the Sysplex do all the work of defining and activating a dynamically movable VIPA (configured via VIPADEFINE and VIPABACKUP) and moving it to another TCP/IP in the event of a failure of the original TCP/IP.

For other types of applications, each application instance must have a unique IP address (they cannot bind to INADDR_ANY), possibly because clients might establish a relationship to that application instance that can span multiple TCP connections. Such server applications bind to the particular IP address, and clients will return to that particular IP address as long as the relationship exists. If the application instance is moved to another Sysplex node on a failure (for example via Automatic Restart Manager), or as a part of a planned application movement across the sysplex, it is important that the IP address (VIPA) is moved with the application instance to the same TCP/IP. This scenario is called the Unique Application Instance Scenario and uses VIPAs that are dynamically allocated via ioctl commands or a BIND.

To keep the Dynamic VIPA with the particular instance, the TCP/IP is dependent on the application to indicate that the Dynamic VIPA is to be activated:

- When the application instance issues a BIND() function call and specifies an IP address that is not active on the TCP/IP, the TCP/IP will activate the address as a Dynamic VIPA, provided it meets certain criteria (see “Overview of Dynamic VIPA Configuration and Operation” on page 81). When the application ends, the Dynamic VIPA is deleted.
- Some applications cannot be configured to issue BIND() for a specific IP address, but nonetheless fall within this scenario. For such applications, a utility is provided (EZBXFDVP) which will cause the designated Dynamic VIPA to be created on behalf of the application. This utility can be included in the JCL or OMVS script which initiates the application. As long as the same JCL package or script is used to restart the application instance on another node in the event of a failure, the same Dynamic VIPA will be activated on the new TCP/IP. The utility may also be used to delete the Dynamic VIPA when the application ends. The application must be APF-Authorized (or the user must have Root authority) to execute the utility

Overview of Dynamic VIPA Configuration and Operation

The following sections describe how to configure Dynamic VIPAs in various scenarios.

Configuration of Dynamic VIPAs

To allow continued and unchanged operation of traditional VIPAs in OS/390 TCP/IP, Dynamic VIPAs are defined with new configuration statements in the PROFILE.TCPIP data set. An overview of the relevant configuration statements is provided below, and see “Chapter 4. Configuring the TCP/IP Address Space” on page 97 for a complete description of the configuration statements. For an example of the configuration statements and display commands for Dynamic VIPA, see “Example of Configuring Dynamic VIPAs” on page 91.

Dynamic VIPAs are defined by statements in the VIPADYNAMIC/ENDVIPADYNAMIC block. There can be more than one VIPADYNAMIC/ENDVIPADYNAMIC block within the profile data set. Within the block, the statements used to define Dynamic VIPAs depend on the application scenario (for example, Multiple Application Instances or Unique Application Instance).

Multiple Application Instance Scenario

For the Multiple Application Instance Scenario, each instance is assigned a unique Dynamic VIPA. The VIPADEFINE configuration statement is used to define the Dynamic VIPA on the TCP/IP where it is normally expected to be active. VIPADEFINE includes a mask to be communicated to the routing daemon as well. When the VIPADEFINE statement is processed in a TCP/IP profile, corresponding DEVICE, LINK, and HOME statements are generated automatically by TCP/IP, and the resulting Virtual device is activated immediately. If OMPROUTE is the associated routing daemon, it is notified immediately. A BSDROUTINGPARMS statement is generated also for orouted.

Because TCP/IP does not know which application is using a Dynamic VIPA, additional configuration is required on other TCP/IPs in the Sysplex to indicate which one should take over the Dynamic VIPA in the event of a failure. The configuration statement used for this purpose is VIPABACKUP. A VIPABACKUP statement does not automatically define and activate a Dynamic VIPA. Instead, it causes the TCP/IP to be ready to take over the Dynamic VIPA when necessary. More than one TCP/IP can be backup for a single Dynamic VIPA, and a rank parameter on the VIPABACKUP statement determines the order of the TCP/IPs for purposes of backup.

Note: If several hosts are configured to back up a Dynamic VIPA with the same rank parameter, timestamps will be used to break the tie. This type of configuration is not recommended, because the ranking order will be indeterminate at configuration time.

The TCP/IPs in the Sysplex exchange information on all VIPADEFINEs and VIPABACKUPs defined in the Sysplex, so that all are aware of which TCP/IP should take over a particular Dynamic VIPA. The list of backup TCP/IPs for any one Dynamic VIPA can be different from the list of backup TCP/IPs for all other Dynamic VIPAs.

Unique Application Instance Scenario

The Unique Application Instance Scenario ties a Dynamic VIPA to a specific instance of an application. However, because errors might occur in configuring applications, TCP/IP needs a mechanism to identify permissible Dynamic VIPAs. This is provided with the VIPARANGE statement. The VIPARANGE statement identifies a range of IP addresses which can be activated as Dynamic VIPAs by an application instance. The VIPARANGE statement consists of a subnet mask and an IP address and thus defines a subnet for Dynamic VIPAs. More than one VIPARANGE statement with different ranges can be defined on a TCP/IP. VIPARANGE does not itself cause any Dynamic VIPAs to be created. Rather, Dynamic VIPAs are created either by an application issuing a BIND() for a specific IP address, or by use of the SIOCSVIPA IOCTL issued by an authorized application or by the EZBXFDVP utility.

When an application issues BIND() for a specific IP address, the receiving TCP/IP checks the IP address against the list of IP addresses in the HOME list. If the IP address has already been defined on this TCP/IP (whether for a physical device, a traditional VIPA, or a Dynamic VIPA) the BIND() execution is successful. If the IP address is not defined on this TCP/IP, the current VIPARANGEs are checked to see if the IP address falls within one of them. If an appropriate VIPARANGE is found and the IP address is not configured elsewhere in the sysplex (as an IP address or Dynamic VIPA), the IP address is activated as a Dynamic VIPA and the operation succeeds. If no apppropriate VIPARANGE is found, or if the IP address is configured elsewhere in the sysplex, the request fails and BIND() returns EADDRNOTAVAIL.

When an authorized application issues the SIOCSVIPA IOCTL to create a Dynamic VIPA, or when the EZBXFDVP utility is executed in JCL or an OMVS script to activate a Dynamic VIPA on behalf of an application instance, the current VIPARANGES are checked to see if the IP address falls within one of them. If an appropriate VIPARANGE is found, and the IP address is not currently defined on this TCP/IP or configured elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the IP address is activated as a Dynamic VIPA. However if no appropriate VIPARANGE is found on this TCP/IP, or if the IP address is currently defined on this TCP/IP or configured elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the request fails with errno and errnojr set to indicate the reason for the failure and the utility ends with a non-zero condition code. Refer to *OS/390 SecureWay Communications Server: IP and SNA Codes* for descriptions of the errnojr values returned.

Note: If the requested IP address has been activated as a Dynamic VIPA by a BIND() or SIOCSVIPA IOCTL elsewhere in the sysplex, an SIOCSIOCTL IOCTL on this TCP/IP will succeed and the already existing Dynamic VIPA will be deleted, and any connections to the deleted VIPA will be terminated.

Using the 'SIOCSVIPA' IOCTL Command

A new ioctl command 'SIOCSVIPA' allows an application to create or delete a Dynamic VIPA on the stack where the application is running. The application issuing the 'SIOCSVIPA' ioctl command must be authorized or running under a user with root authority.

To create a new Dynamic VIPA, the requested IP address must be within a subnet that has been previously specified by a VIPARANGE configuration statement in the

PROFILE.TCPIP data set for this stack. The 'SIOCSVIPA' ioctl command can be used to delete any existing Dynamic VIPA on the stack, whether it was created by a VIPADEFINE or VIPABACKUP configuration statement, a bind(), or a previous 'SIOCSVIPA' ioctl command.

The following example shows how to set up the 'SIOCSVIPA' ioctl command.

```
#include "ezbzdvpc.h"               /* header that contains
                                      the structure for
                                      'SIOCSVIPA' ioctl
                                       and needed constants*/
struct dvreq dv;                    /* the structure passed
                                         on the ioctl command*/
dv.dvr_version = DVR_VER1;           /*version               */
dv.dvr_length = sizeof(struct dvreq);  /* structure length     */
dv.dvr_option = DVR_DEFINE;           /* to define a new
                                         Dynamic VIPA.  Use
                                         DVR_DELETE to delete
                                         a dynamic VIPA        */
dv.dvr_addr.s_addr = inet_addr(my_ipaddr);  /* where my_ipaddr is
                                               a character string
                                               in standard
                                               dotted-decimal
                                               notation */
rc = ioctl(s, SIOCSVIPA, &dv);
```

The 'SIOCSVIPA' ioctl command sets non-zero errno and errnojr values to indicate error conditions. Refer to *OS/390 SecureWay Communications Server: IP and SNA Codes* for a description of the errnojr values returned.

Using the EZBXFDVP Utility

You can use the EZBXFDVP utility to create or delete a Dynamic VIPA. The utility can be initiated from JCL or an OMVS script. EZBXFDVP must be loaded from an authorized library or be invoked from a userid with root authority.

Input Parameters: The input parameters for the utility are:

-p <tcpipname>
Specifies the TCP/IP which is to create or delete a Dynamic VIPA.

-c <IPaddress> or -d <IPaddress>
Specifies to create (-c) or delete (-d) the address (IP address) specified.

Notes:

1. The input parameters -p, -c, and -d must be entered in lower case.
2. <tcpipname> must be entered in upper case.
3. <IPaddress> is dotted-decimal notation.
4. To create a Dynamic VIPA, the specified IP address must be within a subnet that has been previously specified by a VIPARANGE configuration statement in the PROFILE.TCPIP data set for the specified TCP/IP.

Output: The EZBXFDVP utility sets the following exit (completion) codes for create (-c):

0 Success: the Dynamic VIPA was defined/activated for create

4 Warning: the requested Dynamic VIPA was not activated because the specifed IP address is already active on this stack either as a VIPA or a real IP address.

8 Error: the requested Dynamic VIPA was not defined/activated.

The EZBXFDVP utility sets the following exit (completion) codes for delete (-d):

0 Success: the Dynamic VIPA was deleted

8 Error: the specified IP address was not defined as a Dynamic VIPA on this TCP/IP.

Notes:

1. When an error is detected, the ernno text and errnojr value are printed to stderr.
2. If the IP address requested for the Dynamic VIPA is not within a VIPARANGE configured on this stack, completion code 8 is returned even if the IP address is currently active on this stack

Examples

Within JCL:

```
//TCPDVP EXEC  PGM=EZBXFDVP,REGION=0K,TIME=1440,                        X
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS3 -c 1.2.3.4'
```

From OMVS shell:

```
EZBXFDVP -p TCPCS3 -c 1.2.3.4
```

Planning for the Multiple Application Instance Scenario

In the Multiple Application Instance Scenario, instances of the application in question are allocated among Sysplex nodes according to some plan, presumably related to balancing workload across available capacity. This allocation is done independently of VIPA Takeover and must occur before planning the corresponding VIPA Takeover strategy. After the desired allocation of application instances to Sysplex nodes has been performed, the considerations for the associated Dynamic VIPAs are as follows:

1. For each instance of a particular application to be supported via Dynamic VIPA, allocate a unique Dynamic VIPA address, and add a VIPADEFINE statement to the TCP/IP profile for the TCP/IP associated with the application instance.
2. For each of the above Dynamic VIPAs, determine which application instance or instances should take over the workload, considering probable capacity and any other application-related considerations.

   Note: In the event of failure of a TPC/IP where one of the application instances is running, the associated Dynamic VIPA and all subsequent corresponding client work load will be moved to one and only one other TCP/IP.

   If more than one TCP/IP is to provide backup for a Dynamic VIPA (to allow for a failure of a production environment during a planned outage of another TCP/IP, for example), determine the order in which the selected TCP/IPs should be designated as backup. Add a VIPABACKUP statement to each TCP/IP which is to provide backup for the Dynamic VIPA, with appropriate rank values to determine the order. Do this for each of the Dynamic VIPAs in the above step.
3. Perform steps 1 and 2 for each other application to be supported by Dynamic VIPAs. In allocating Dynamic VIPAs and backups, being careful to ensure that no failure scenario results in more than 64 active and backup Dynamic VIPAs on any one TCP/IP.

Note: It is possible to share a Dynamic VIPA among several different applications, but in doing so, ensure that instances of all such applications will exist on any TCP/IP to which the Dynamic VIPA may be moved in case of a failure.

After these steps are complete, start the affected TCP/IPs (or modify their configuration via VARY OBEY), provide DNS configuration for the application names as appropriate, and start the application instances. From that point on, the TCP/IPs in the Sysplex will collaborate to ensure that each Dynamic VIPA is kept active somewhere within the Sysplex as long as there is at least one functioning TCP/IP which has been designated as backup for the Dynamic VIPA.

Planning for the Unique Application Instance Scenario

In the Unique Application Instance Scenario, each application instance is assigned an unique IP address as its Dynamic VIPA. Before allocating individual Dynamic VIPAs, a block or blocks of IP addresses should be defined for these Dynamic VIPAs, and the individual Dynamic VIPAs should be allocated from within the block or blocks. Each block should be represented as a subnet, so that a VIPARANGE statement can be defined for the block.

Follow these steps when setting up any unique application instances:

1. For each application instance, assign a Dynamic VIPA from one of the blocks of IP addresses for this purpose. Do not allocate an IP address which is also assigned to another application instance, or which is defined by VIPADEFINE for the Multiple Application Instance Scenario. Configure the application to use this Dynamic VIPA (if it issues BIND() for a particular IP address), or add the EZBXFDVP utility to the JCL or OMVS script and configure the EZBXFDVP utility to activate the Dynamic VIPA before starting the application, and to delete the Dynamic VIPA when the application ends.
2. For each application instance, determine on which TCP/IP the application instance will normally be executed and to which TCP/IP(s) the application instance could be moved in case of failure of the normal TCP/IP or the application itself. For each such TCP/IP, add to the profile a VIPARANGE statement specifying a range for which the selected Dynamic VIPA is valid.

   Note: Within each range subnet, the broadcast address and the net prefix cannot be used a Dynamic VIPAs.
3. Perform steps 1 and 2 until all application instances have been allocated a unique Dynamic VIPA.

The application restart strategy should ensure that the worst-case failure scenario does not attempt to create more than 64 Dynamic VIPAs on a single TCP/IP. If such an attempt is made, creation of the 65th Dynamic VIPA will fail, with possible resulting loss of connectivity from clients to the server application.

Note: The limit of 64 Dynamic VIPAs on a single TCP/IP applies to all Dynamic VIPAs, whether created by VIPADEFINE/ VIPABACKUP configuration statements, by a BIND() call, or by executing the EZBXFDVP utility.

Allocating a single block makes the definition process easier, but also provides less individual control. Alternatively, because the smallest subnet consists of four IP addresses, defining a unique subnet for each Dynamic VIPA in this scenario *wastes* three other IP addresses that could have been used for Dynamic VIPAs.

Choosing Which Form of Dynamic VIPA Support to Use

The following sections explain which of the new features should be used for the type of application being used.

When should VIPADEFINE and VIPABACKUP be used to define a Dynamic VIPA?

- Application(s) bind to INADDR_ANY and exist on multiple TCP/IPs.
- Dynamic VIPA takeover is desired.
- The VIPA does not need to be deleted when the application is stopped.

When should VIPARANGE and bind() be used to define a Dynamic VIPA?

- The application cannot bind to INADDR_ANY or Dynamic VIPA takeover is not desired.
- The IP address to which the application binds can be controlled by the user.
- Automatic deletion of the Dynamic VIPA when the application is stopped is acceptable.
- A specific Dynamic VIPA address must be associated with a specific application.
- The application is not APF authorized, or not run under a userid with Root authority.

When should VIPARANGE and the EZBXFDVP utility (or ioctl command 'SIOCSVIPA') be used to define a Dynamic VIPA?

- The application cannot bind to INADDR_ANY or Dynamic VIPA takeover is not desired.
- The IP address to which the application binds is known but cannot be controlled by the user.
- Automatic deletion of the Dynamic VIPA when the application is stopped is not acceptable.
- The EZBXFDVP utility (or application issuing the ioctl command) will be run from an APF authorized library or under a userid with Root authority.

Resolution of Dynamic VIPA Conflicts

When a TCP/IP has an active Dynamic VIPA, and the TCP/IP fails, the Dynamic VIPA is no longer active in the Sysplex. Similarly, when the Dynamic VIPA is activated via BIND() from an application and the application ends, the Dynamic VIPA is likewise deleted. However, an attempt might be made to activate the same Dynamic VIPA on two different TCP/IPs in the Sysplex. Because the routers cannot handle this situation reliably, the TCP/IPs in the Sysplex will collaborate to ensure that this does not happen.

Dynamic VIPA Creation Results

Table 7 on page 87 summarizes the results of attempting to create a Dynamic VIPA when it (or an IP address for HOME statement) is already defined in the sysplex.

Table 7. Summary of Dynamic VIPA Creation Results

First Action	Second Action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
bind()	bind()	Second bind() succeeds, but no new VIPA is created.	Second bind() fails.
bind()	ioctl()	ioctl() fails with warning condition code, but the application associated with the ioctl is still able to use the Dynamic VIPA.	ioctl() succeeds, bind is deleted.
bind()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
bind()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
bind()	HOME	See note.	See note.
ioctl()	bind()	bind() succeeds, no new VIPA is created.	bind() fails.
ioctl()	ioctl()	Second ioctl() fails.	Second ioctl() succeeds.
ioctl()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
ioctl()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
ioctl()	HOME	See note.	See note.
VIPADEFINE	bind()	bind() succeeds, but no new VIPA is created.	bind() fails.
VIPADEFINE	ioctl()	ioctl() fails.	ioctl() fails.
VIPADEFINE	VIPADEFINE	VIPADEFINE fails if the VIPADEFINE statement specifies a different mask than the first VIPADEFINE specified. If the second VIPADEFINE statement is an exact duplicate of the first, the second VIPADEFINE is ignored with no error message.	Second VIPADEFINE activation is deferred until there are no connections on stack one, at which point, stack one reverts to backup status.
VIPADEFINE	VIPABACKUP	VIPABACKUP fails.	Both succeed.
VIPADEFINE	HOME	See note.	See note.
VIPABACKUP	bind()	bind() fails.	If the IP address is already active on the bind() stack, the bind() will succeed. Otherwise, the bind() fails.
VIPABACKUP	ioctl()	ioctl() fails.	ioctl() fails.
VIPABACKUP	VIPADEFINE	VIPADEFINE succeeds, replaces the VIPABACKUP.	VIPADEFINE succeeds.
VIPABACKUP	VIPABACKUP	Second VIPABACKUP succeeds.	Second VIPABACKUP succeeds.
VIPABACKUP	HOME	See note.	See note.
HOME	bind()	bind() succeeds, but no new VIPA is created.	bind() fails, but no new VIPA is created.
HOME	ioctl()	ioctl() fails.	ioctl() fails.
HOME	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.

Table 7. Summary of Dynamic VIPA Creation Results (continued)

First Action	Second Action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
HOME	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
Note: Defining the same IP address in the HOME statement as an existing Dynamic VIPA will not be rejected by the TCP/IP stack, but it is likely to cause routing problems.			

The following sections illustrate what happens when such an attempt is made.

Restart of the Original VIPADEFINE TCP/IP After a Failure

When a Dynamic VIPA is defined via VIPADEFINE on one TCP/IP, and other TCP/IPs are designated as backup via VIPABACKUP statements for the same Dynamic VIPA, one of the backup TCP/IPs will activate the Dynamic VIPA if the VIPADEFINE TCP/IP fails. When that TCP/IP is restarted, it is not initially aware that a failure has occurred. At the same time, the application might have established connections via the Dynamic VIPA on the backup TCP/IP. If the VIPADEFINE TCP/IP immediately takes the Dynamic VIPA back, connections to the application on the backup TCP/IP will be terminated, causing disruption to the clients. Because this might not be acceptable to customers, and because TCP/IP cannot tell when this is the case, the following occurs in the restart scenario:

- When the TCP/IP with the VIPADEFINE is restarted, the TCP/IP exchanges information with the other TCP/IPs in the Sysplex.
- If there are no active connections to the Dynamic VIPA on the backup TCP/IP, the backup TCP/IP deletes the Dynamic VIPA and reverts to backup status in its proper order in the backup list for the Dynamic VIPA (normally first in the list). The VIPADEFINE TCP/IP activates the Dynamic VIPA as usual, and it is now ready to receive new work for that Dynamic VIPA. A small window exists between when the check is made and when the Dynamic VIPA is moved. If connections are made to the old host in this interval, they will be broken when the VIPA is activated on the new host.
- If there are active connections to the Dynamic VIPA on the backup TCP/IP, the backup TCP/IP notifies the restarting TCP/IP, and the restarting TCP/IP does not activate the Dynamic VIPA. Instead, the restarting TCP/IP is placed first in the backup list for that Dynamic VIPA, ahead of all VIPABACKUP TCP/IPs, while the backup TCP/IP periodically queries its status. When it appears that there are no more connections to the Dynamic VIPA on the backup TCP/IP, the backup TCP/IP deletes the Dynamic VIPA and returns to its normal place in the backup list. Because the restarted VIPADEFINE TCP/IP is first in the backup list, the restarted TCP/IP can now activate the Dynamic VIPA as originally intended.

While the Dynamic VIPA is active on the backup TCP/IP, it will continue to accept new TCP connections to that VIPA. TCP/IP does not implement any kind of quiesce function on the Dynamic VIPA, because this would mean refusing new client requests, which is the same as an outage. If moving the work back to the VIPADEFINE TCP/IP is more important than maintaining uninterrupted service to all clients, the system operator can use VARY OBEY to delete the Dynamic VIPA on the backup TCP/IP with the VIPADELETE profile statement, which will cause immediate takeover by the VIPADEFINE TCP/IP. The operator can then restore the backup TCP/IP to backup status by VARY OBEY with the original VIPABACKUP statement. Though this reversal is a manual operation (which may be automated if desired), it allows the installation to make the decision on whether immediate reversal or uninterrupted service is more important.

Movement of Unique Application Instance (BIND)

When two different application instances attempt to activate a Dynamic VIPA within a valid VIPARANGE via BIND() on different TCP/IP stacks, this might be due to an error in configuring one of the instances or a real desire to move the instance to a new TCP/IP. Because TCP/IP cannot determine the difference, the first application instance will succeed in activating the Dynamic VIPA, and the second one will fail, under the assumption that it is a configuration error. Again, it is not possible to have the same Dynamic VIPA active at the same time on two or more TCP/IPs. If movement of the application instance from one TCP/IP to another is intended, the first application instance must be ended on the first TCP/IP before it is started on another TCP/IP.

Movement of Unique APF-Authorized Application Instance (Utility-activated)

The supplied EZBXFDVP utility makes use of IOCTL() to activate a Dynamic VIPA. Because this happens independently of any application, a decision was made that this should only occur with special care, so the EZBXFDVP utility must be executed with an application that is APF-authorized or by a user having Root authority. Because this is a controlled environment, it is assumed that configuration errors are minimized or avoided. Therefore, TCP/IP assumes that this controlled use is always correct. If the EZBXFDVP utility (or an APF-authorized or Root authority application programmed to use the IOCTL() interface) requests a Dynamic VIPA that was previously activated on another TCP/IP via BIND() or the EZBXFDVP utility, the second request will succeed, and the existing Dynamic VIPA will be deleted on the other TCP/IP, even if this means that TCP connections are terminated on the other TCP/IP.

Same Dynamic VIPA as VIPADEFINE and BIND()/UTILITY/IOCTL()

Regardless of careful implementation, it is possible that the same IP address is inadvertently selected for VIPADEFINE and for use with BIND() or the EZBXFDVP utility. Because the application scenarios are quite different, this must be an error.

If this occurs on the same TCP/IP, the second attempt may fail. (If bind() occurs after VIPADEFINE, bind() will succeed.)In other words, if an IP address is specified in a VIPADEFINE and that same IP address has already been activated on the TCP/IP by an application via BIND() or the EZBXFDVP utility, the VIPADEFINE will be rejected during VARY OBEY profile processing. If an IP address is activated via VIPADEFINE, and application does a BIND() or IOCTL() or the EZBXFDVP utility, the BIND() will succeed, but the IOCTL() will fail with a non-zero errno and the EZBXFDVP utility will set a non-zero condition.

The same situation could also occur on two different TCP/IPs in the Sysplex. Because the TCP/IPs are exchanging information among themselves, if the two attempts are far enough apart in time, the second attempt will be caught immediately. However, it is possible that the attempt will be made almost simultaneously on two different TCP/IPs, such that neither TCP/IP is yet aware of the attempt on the other TCP/IP. If both attempt such an activation, and the exchange of information then shows a conflict, the internal Sysplex time stamps are used to determine which attempt was really first. The one that appears to be first is allowed to continue, and the Dynamic VIPA is deleted from the *later* TCP/IP. While such a simultaneous attempt is somewhat unpredictable in respect to which one will

succeed, the Dynamic VIPA will remain active on only one TCP/IP, and examination of messages will indicate the configuration error to be corrected.

Other Considerations

The following sections describe other considerations you should understand regarding Dynamic VIPA support.

Mixture of Types of Dynamic VIPAs within Subnets

Any particular IP address can be used in only one way as a Dynamic VIPA. As described in previous sections, a Dynamic VIPA can be defined either via VIPADEFINE or by application action within a valid VIPARANGE, but not both. However, within a subnet defined as a VIPARANGE, some IP addresses can be used for VIPADEFINE, and others may be assigned to unique application instances, without conflict, as long as the limit of a total of 64 active and backupDynamic VIPAs on a single TCP/IP is not exceeded. TCP/IP will make no attempt to reject a VIPADEFINE Dynamic VIPA that also falls within a VIPARANGE. This allows installations with limited availability of IP addresses to assign individual addresses to either application scenario, without having to define separate subnets and use up additional IP addresses in that manner.

MVS Failure and Sysplex Failure Management

The TCP/IPs in a Sysplex use MVS XCF Messaging to exchange information about Dynamic VIPAs. When a TCP/IP fails or is ended by operator command, but the underlying MVS remains active, notification of the other TCP/IPs is immediate, and takeover of VIPADEFINE Dynamic VIPAs is automated and very fast.

However, when an MVS fails, there is normally an operator message on the console requiring a response (WTOR). Until this response is made by an operator or automation, the other MVSs do not notify the remaining TCP/IPs in the Sysplex of the failure of the TCP/IP on the failing MVS. This can delay automated backup of VIPADEFINE Dynamic VIPAs. Sysplex Failure Management (SFM) can be used to automate the required response to the console message of the failing MVS. Refer to *OS/390 MVS Setting Up a Sysplex* for information on how to set up SFM to avoid the requirement for a manual response and speed backup of VIPADEFINE Dynamic VIPAs.

UDP and Dynamic VIPAs

While most applications support multiple instances in a Sysplex, very few applications expect IP addresses to move around under them. TCP applications use TCP connections to form a relationship between particular client and server instances to exchange data over an extended period and rely on notification of TCP connection termination to initiate recovery and to reestablish a new relationship (possibly from a client to a different server). Conversely, most UDP applications do the equivalent function at the application layer. Movement of an IP address to a different server could be confusing to the client, unless the new server also is aware of the state of the client work.

UDP applications whose interactions consist of atomic interactions (a single request followed by one or more responses, with no state information maintained at the server between requests) can utilize Dynamic VIPAs in the Multiple Application

Instance Scenario. However, if the server application maintains state information between interactions (for example, NFS), then moving a Dynamic VIPA to another server might not work unless the client/server application protocol can detect the discontinuity. In that case, the Unique Application Instance Scenario might apply, which would require the restart of the server instance on another TCP/IP. When considering Dynamic VIPAs for use with UDP applications, carefully investigate the application to determine whether it will work in one scenario or the other, or if it should continue to use traditional VIPAs.

Example of Configuring Dynamic VIPAs

The TCP/IP profiles needed to implement Dynamic VIPA (DVIPA) on multiple systems in a sysplex are shown in the following examples. The VIPADEFINE and VIPABACKUP statements allow VIPA Takeover to occur if needed (see “Multiple Application Instance Scenario” on page 81) and the VIPARANGE statements allow Dynamic VIPAs to be dynamically created by an application or by the EZBXFDVP utility (see “Planning for the Unique Application Instance Scenario” on page 85).

```
; TCPCS
VIPADYNAMIC
VIPADEFINE      255.255.255.192   201.2.10.11    201.2.10.12
VIPABACKUP 100                    201.2.10.13
VIPABACKUP 80                     201.2.10.21    201.2.10.22
VIPARANGE DEFINE 255.255.255.192  201.2.10.192
ENDVIPADYNAMIC
```

```
; TCPCS2
VIPADYNAMIC
VIPADEF          255.255.255.192   201.2.10.13
VIPAB 100                          201.2.10.11    201.2.10.21
VIPAB 75                           201.2.10.12    201.2.10.22
VIPARANGE DEFINE 255.255.255.192   201.2.10.192
ENDVIPADYNAMIC
```

```
; TCPCS3
VIPADYNAMIC
VIPADEF           255.255.255.192 201.2.10.21     201.2.10.22
VIPAB 10                          201.2.10.21.11  201.2.10.12  201.2.10.13
VIPARANGE DEFINE 255.255.255.192  201.2.10.192
ENDVIPADYNAMIC
```

```
; TCPCS6
;Does not contain a VIPADYNAMIC section
```

Start TCP/IP on each system as shown above.

- On system1, start TCPCS and TCPCS2.
- On system2, start TCPCS3, on system3 start TCPCS6.
- On system1, run the EZBXFDVP utility to define the DVIPA 201.2.10.193.

```
//TCPDVP   PROC
//*
//*
//TCPDVP  EXEC PGM=EZBXFDVP,REGION=0K,TIME=1440,                        x
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS -c 201.2.10.193'
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSERR   DD SYSOUT=*
//SYSERROR DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//SYSUDUMP DD SYSOUT=A
//SYSABEND DD SYSOUT=*
//*
//*Run program here
```

```
//*
//TCPDVP  EXEC PGM=EZBXFDVP,REGION=0K,TIME=1440,                      x
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS -d 201.2.10.193'
```

The PARM field can be -c for create or -d for delete. The example above will create DVIPA 201.2.10.193 for the TCP/IP named TCPCS. After intermediate program has completed, the DVIPA will be deleted.

Verifying the Configuration

A display command parameter displays Dynamic VIPAs in the sysplex. In the following example for stack TCPCS, the ORIGIN lines show that 201.2.10.11 and 201.2.10.12 were created by VIPADEFINE on this stack, 201.2.10.193 was created by VIPARANGE IOCTL (issued through the EZBXFDVP utility), and all of the others were created by VIPABACKUP statements.The command is:

```
d tcpip,tcpname,sysplex,vipadyn
```

The status (STATUS) of each DVIPA on the stack (TCPNAME) and system (MVSNAME) in the sysplex is shown in this display. The rank (RANK) indicates which of the stacks is eligible to take over if the stack on which the DVIPA is active and can be stopped will be chosen. The stack with the highest rank is the one that will take over the DVIPA. If the DVIPA is configured on the stack specified in the display command, an ORIGIN line appears to indicate how the DVIPA is configured on the specified stack.

```
D TCPIP,TCPCS,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R8 498
VIPA DYNAMIC DISPLAY FROM TCPCS    AT SYSTEM1
IPADDR: 201.2.10.11  LINKNAME: VIPLC9020A0B
  ORIGIN: VIPADEFINE
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.12  LINKNAME: VIPLC9020A0C
  ORIGIN: VIPADEFINE
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  -----------      ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  075
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.13
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS2    SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.21
   ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS     SYSTEM1   BACKUP  080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  080
  TCPCS2    SYSTEM1   BACKUP  075
```

```
IPADDR: 201.2.10.193  LINKNAME: VIPLC9020AC1
  ORIGIN: VIPARANGE IOCTL
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.192
16 OF 16 RECORDS DISPLAYED
```

TCPCS2 displays the same information about all the DVIPAs. ORIGIN fields are displayed for the DVIPAs that are configured on this stack.

```
D TCPIP,TCPCS2,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R8 504
VIPA DYNAMIC DISPLAY FROM TCPCS2   AT SYSTEM1
IPADDR: 201.2.10.11
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.12
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  075
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.13  LINKNAME: VIPLC9020A0D
  ORIGIN: VIPADEFINE
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS2    SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS     SYSTEM1   BACKUP  080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  080
  TCPCS2    SYSTEM1   BACKUP  075
IPADDR: 201.2.10.193
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  --------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.192
16 OF 16 RECORDS DISPLAYED
```

In the following example, TCPS6 on SYSTEM3 knows about the Dynamic VIPAs on the other stacks. There are no Dynamic VIPAs configured on TCPCS6, so there are no ORIGIN fields displayed.

```
D TCPIP,TCPCS6,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R8 495
VIPA DYNAMIC DISPLAY FROM TCPCS6   AT SYSTEM3
IPADDR: 201.2.10.11
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
```

```
IPADDR: 201.2.10.12
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  075
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.13
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS2    SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.21
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS     SYSTEM1   BACKUP  080
IPADDR: 201.2.10.22
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  080
  TCPCS2    SYSTEM1   BACKUP  075
IPADDR: 201.2.10.193
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.192
16 OF 16 RECORDS DISPLAYED
```

NETSTAT Support for Dynamic VIPAs

The netstat commands (TSO NETSTAT, OS/390 UNIX onetstat, and MVS console D TCPIP,,Netstat have a new section in the CONFIG (-f) report and a new VIPADYN (-v) report. These reports show the Dynamic VIPA configuration for a particular TCP/IP.

The new Global Configuration Information section of the CONFIG (-f) report is displayed whether there are any DVIPAs known to this stack or not. The Dynamic VIPA Information section is only displayed when there are DVIPAs configured on this stack. The VIPA Range section, displayed only if a VIPARANGE statement was processed in this stack's profile (or obeyfile), indicates only that a range was configured, it does not indicate whether any IOCTL or BIND has actually created a DVIPA in the specified range.

On stack, TCPCS:

```
Dynamic VIPA Information:
  VIPA Backup:
    IP Address         Rank
    ----------         ----
    201.2.10.13        000100
    201.2.10.21        000080
    201.2.10.22        000080
  VIPA Define:
    IP Address       AddressMask
    ----------       -----------
    201.2.10.11      255.255.255.192
    201.2.10.12      255.255.255.192
  VIPA Range:
    AddressMask      IP Address
    -----------      ----------
    255.255.255.192  201.2.10.192
```

On stack, TCPCS2:

```
Dynamic VIPA Information:
  VIPA Backup:
    IP Address       Rank
    ----------       ----
    201.2.10.11      000100
    201.2.10.12      000075
    201.2.10.21      000100
    201.2.10.22      000075
  VIPA Define:
    IP Address       AddressMask
    ----------       -----------
    201.2.10.13      255.255.255.192
  VIPA Range:
    AddressMask      IP Address
    -----------      ----------
    255.255.255.192  201.2.10.192
```

The new VIPADYN (-v) report displays all the Dynamic VIPAs available to this stack, as shown in the following example.

```
On stack TCPCS:

MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPCS
IP Address       AddressMask      Status  Origination
----------       -----------      ------  -----------
201.2.10.11      255.255.255.192  Active  VIPADefine
201.2.10.12      255.255.255.192  Active  VIPADefine
201.2.10.13      255.255.255.192  Backup  VIPABackup
201.2.10.21      255.255.255.192  Backup  VIPABackup
201.2.10.22      255.255.255.192  Backup  VIPABackup
201.2.10.193     255.255.255.192  Active  VIPARange IOCTL
On stack TCPCS2:

MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPCS2
IP Address       AddressMask      Status  Origination
----------       -----------      ------  -----------
201.2.10.11      255.255.255.192  Backup  VIPABackup
201.2.10.12      255.255.255.192  Backup  VIPABackup
201.2.10.13      255.255.255.192  Active  VIPADefine
201.2.10.21      255.255.255.192  Backup  VIPABackup
201.2.10.22      255.255.255.192  Backup  VIPABackup
On stack TCPCS6:

MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPCS6
IP Address       AddressMask      Status  Origination
----------       -----------      ------  -----------
```

Dynamic VIPAs and Routing Protocols

OROUTED

The Dynamic VIPA Support generates the appropriate BSDROUTINGPARMS statements for OROUTED to support Dynamic VIPAs.

If using RIP services (OMPROUTE, OROUTED) and Host Route Broadcasting is not supported by adjacent routers (that is, inability to learn host routes), the following restrictions for VIPA addresses must be applied to benefit from fault tolerance support:

- If you use subnetting and VIPA addresses are in the same network as the physical IP addresses, the subnetwork portion of any VIPA addresses must not

be the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the VIPA address.

- If subnetting is not used on any physical interface, the network portion of any VIPA addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the VIPA address, preferably a class C network address.

If using RIP services (OMPROUTE, OROUTED) and Host Route Broadcasting is supported by adjacent routers, the network or subnetwork portions of VIPA addresses can be the same across multiple OS/390 TCP/IP stacks in the network. See the OMPROUTE or OROUTED topics for more information on how to configure Host Route Broadcasting on the RIP services.

OMPROUTE

The names of Dynamic VIPA interfaces are assigned dynamically by the stack when a Dynamic VIPA interface is created. Therefore, the Name field set on the Interface or OSPF_Interface statement for a Dynamic VIPA will be ignored by OMPROUTE.

It is recommended that a host have an Interface or OSPF_Interface definition for every Dynamic VIPA address which that host might own. Because this could be a large number of interfaces, additional wildcard capabilities are added to OMPROUTE, for Dynamic VIPA interfaces only.

Ranges of Dynamic VIPA interfaces can be defined using the subnet mask parameter on the OSPF_Interface or Interface statement. The range defined will be all the IP addresses that fall within the subnet defined by the mask and the IP address. For example:

```
OSPF_Interface
   IP_address = 10.138.165.80
   Name = dummy_name (see note)
   Subnet_mask = 255.255.255.240;
```

Note: The *Name* parameter is required, but it is not actually used for Dynamic VIPAs.

Defines a range of Dynamic VIPA addresses from 10.138.165.80 to 10.138.165.95.

For consistency with the VIPARANGE statement in the TCP/IP profile, any value that may fall within the range can be used with the mask to define a range of Dynamic VIPAs. The interface statement in the following example has the same meaning as the one in the example above:

```
OSPF_Interface
   IP_address = 10.138.165.87
   Name = dummy_name
   Subnet_mask = 255.255.255.240;
```

Notes:

1. When defining ranges, it is not necessary or desirable to code a destination address. OMPROUTE will automatically set the destination address of a Dynamic VIPA to its IP address.
2. There is nothing in the interface definition statements that informs OMPROUTE that a particular interface definition statement is for a Dynamic VIPA or a range of Dynamic VIPAs. Rather, OMPROUTE learns this information from the stack when these interfaces are created or taken over.

Chapter 4. Configuring the TCP/IP Address Space

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

When TCP/IP is started, it loads configuration information from a profile data set. This data set is referred to as *hlq*.PROFILE.TCPIP or PROFILE.TCPIP in this book. This chapter describes how to configure the TCP/IP address space by updating the configuration statements in this data set. It also describes how you can dynamically change the configuration for the current session with the VARY TCPIP command.

The following topics are included in this chapter:

- Update the TCP/IP cataloged procedure
- Specify configuration statements in PROFILE.TCPIP
- PROFILE.TCPIP configuration statements
- TCP/IP address space configuration statements (listed in alphabetical order)

Note

You can modify many of the configuration statements; this is described in the Modifying section for each statement. For more information about modifying statements, see “VARY TCPIP,,OBEYFILE” on page 267 also. In addition to modifying information, each statement description includes a Usage Notes section. Read this section for important configuration information.

Note: CS for OS/390 exploits OS/390 UNIX services even for traditional MVS environments and applications. Prior to utilizing TCP/IP services, therefore, an OS/390 UNIX environment — including DFSMS™, a Hierarchical File System, and a security product (such as RACF) — needs to be defined and active.

Update the TCP/IP Cataloged Procedure

Copy the TCP/IP cataloged procedure in *hlq*.SEZAINST(TCPIPROC) to your system or recognized PROCLIB and modify it to suit your local conditions. Specify TCP/IP parameters and remove or change the DD statements as required. The jobname associated with the started task of the TCP/IP system address space must match the NAME parameter on the SUBFILESYSTYPE statement in the BPXPRMxx member of ’SYS1.PARMLIB’ used to start OS/390 UNIX. For more information on BPXPRMxx, see the *OS/390 UNIX Planning*. Also see *OS/390 MVS Initialization and Tuning Reference*.

TCP/IP Cataloged Procedure (TCPIPROC)

The following is an example of a TCP/IP Cataloged Procedure that defines the component tracing that is to be in effect for the TCP/IP address space. CTIEZB00 indicates the component trace member of SYS1.PARMLIB.

```
//TCPIP    PROC PARMS = 'CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//*  SMP/E Distribution Name: EZAEB01G
//*
//*  Licensed Materials - Property of IBM
//*  "Restricted Materials of IBM"
//*  5647-A01
//*  (C) Copyright IBM Corp. 1991, 1999
//*  Status = CSV2R8
//*
//TCPIP    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM='&PARMS'
//*            PARM=('&PARMS',
//*        'ENVAR("RESOLVER_CONFIG=//''TCPIVP.TCPPARMS(TCPDATA)''")')
//*
//*     The C runtime libraries should be in the system's link list
//*     or add them via a STEPLIB definition here.  If you add
//*     them via a STEPLIB, they must be APF authorized with DISP=SHR
//*
//*STEPLIB  DD ...
//*
//*     SYSPRINT contains run-time diagnostics from TCPIP. It contains
//*        trace resolver output and TCPIPSTATISTICS output.
//*        It can be directed to SYSOUT or a data set.  We recommend
//*        directing the output to SYSOUT due to data set size
//*        restraints.
//*     ALGPRINT contains run-time diagnostics from TCPIP's Autolog
//*        task. It can be directed to SYSOUT or a data set.  We
//*        recommend directing the output to SYSOUT due to data set size
//*        restraints.
//*     CFGPRINT contains run-time diagnostics from TCPIP's Config
//*        task. It also contains run-time diagnostics from
//*        D TCPIP,,NETSTAT,,DEBUG and D TCPIP,,SYSPLEX,,DEBUG
//*        commands.  It can be directed to SYSOUT or a data set.  We
//*        recommend directing the output to SYSOUT due to data set size
//*        restraints.
//*     SYSERROR contains error messages from TCPIP that occurred
//*        while processing the PROFILE.
//*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//*
//*        TNDBCSCN is the configuration data set for TELNET DBCS
//*        transform mode.
//*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*
//*        TNDBCSXL contains binary DBCS translation table codefiles
//*        used by TELNET DBCS Transform mode.
//*
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*
//*        TNDBCSER receives debug output from TELNET DBCS Transform
//*        mode, when TRACE TELNET is specified in the PROFILE data set.
//*
//*TNDBCSER DD SYSOUT=*
//*
//*     TCPIP reads the parameters from a data set with name
//*     TCPIP.nodename.TCPIP or with name TCPIP.PROFILE.TCPIP.
//*     See the chapter on "Configuring the TCPIP Address Space" in
//*     the Configuration Guide for more information. A sample of
//*     such a profile is included in member SAMOPROF of the
```

```
//*     SEZAINST data set.
//*
//*PROFILE  DD DISP=SHR,DSN=TCPIVP.TCPPARMS(PROFILE)
//*PROFILE  DD DISP=SHR,DSN=TCPIP.PROFILE.TCPIP
//*
//*     SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*     For more information please see "Understanding TCP/IP Data
//*     Set Names" in the Configuration Guide.
//*
//*SYSTCPD  DD DSN=TCPIVP.TCPPARMS(TCPDATA),DISP=SHR
//*SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Using Output Data Sets

In the TCP/IP address space, the SYSPRINT and SYSERROR data sets defined with a DD statement must have a variable blocked (VB) format. Block size (BLKSIZE) for a VB RECFM must be at least four bytes larger than the logical record length (LRECL). You can allocate these as partitioned or sequential data sets, but be aware that partitioned data sets cannot be reused if they have filled or if the members already exist.

Output to these data sets is handled in the following manner:

- If only a primary data set is defined, the primary data set is overwritten.
- If any of the alternate data sets are specified, initially the primary data set is overwritten. Should this data set fill, it is closed and output is sent to the next alternate data set. Existing data in the alternate data sets is overwritten. This process continues until all defined data sets have been filled. Once this occurs, the primary data set is overwritten again and the cycle continues.

Specifying TCP/IP Address Space Parameters

Specify the CTRACE parameter on the PARMS=CTRACE(CTIEZB00) keyword of the EXEC JCL statement in the TCP/IP started procedure to indicate the member of the SYS1.PARMLIB to be used by TCP/IP Component Trace processing. A sample of this parmlib member can be found in SYS1.PARMLIB. You can find a description of the TCP/IP Component Trace support in *OS/390 SecureWay Communications Server: IP Diagnosis*.

Specify the ENVAR parameter on the PARMS= keyword to override the resolver file. For more information on setting the environment variable RESOLVER_CONFIG using the ENVAR parameter, see “Considerations for Multiple Instances of TCP/IP” on page 57.

Specify Configuration Statements in PROFILE.TCPIP

During initialization of the TCP/IP address space, system operation and configuration parameters are read from a configuration profile data set.

A sample of this data set, SEZAINST(SAMPPROF), is shown in “Sample Profile Configuration Data Set (SAMPPROF)” on page 105. You can copy and modify this sample to use as your default configuration profile.

As you create and name your configuration profile, be aware of the search order TCP/IP uses to find this data set:

1. //PROFILE DD DSN=

   TCP/IP looks first for a data set specified by the //PROFILE DD statement in the cataloged procedure.
2. *job_name*.*node_name*.TCPIP

   If there is no //PROFILE statement, TCP/IP looks next for *job_name*.*node_name*.TCPIP, where *node_name* is the node name returned from __ipnode() (the node name specified on the VMCF initialization record). For more information on __ipnode(), see the *OS/390 C/C++ Run-Time Library Reference*.
3. If this data set is not found, the program uses the first of the following data sets it finds:
   a. TCPIP.*node_name*.TCPIP
   b. *job_name*.PROFILE.TCPIP
   c. TCPIP.PROFILE.TCPIP

To customize your system, specify system operation parameters and network configuration information in this data set using the configuration statements listed in Table 9 on page 124. You can find the complete statement syntax and descriptions in alphabetical order in “PROFILE.TCPIP Configuration Statements” on page 119.

You can also put many of these statements in a separate data set, process it with the VARY TCPIP command, and dynamically change the TCP/IP configuration established by the PROFILE.TCPIP data set. For more information, see “VARY Command—TCPIP Address Space” on page 264.

MVS system symbols (such as &SYSCLONE, &SYSNAME, and &SYSPLEX) can be used in the PROFILE.TCPIP and OBEYFILE data sets. Thus, several TCP/IP systems can share one PROFILE.TCPIP file, reducing the number of those data sets that must be maintained. TCP/IP translates the symbols as it reads the file. For detailed information about symbols and how to define them, refer to the *OS/390 MVS Initialization and Tuning Reference*.

Changing Configuration Information

You can dynamically change many of the TCP/IP configuration options established by the PROFILE.TCPIP data set. You can put the changed configuration statements in a separate data set and process it with the VARY TCPIP command. For more information about VARY TCPIP, see “VARY Command—TCPIP Address Space” on page 264.

See the Modifying section in each configuration statement explanation for a description of how to dynamically change the information for that statement.

Note: The FIREWALL option on the IPCONFIG statement cannot be changed dynamically. To change the status of FIREWALL support, the TCP/IP stack must be recycled.

DEVICE and LINK Statements

CS for OS/390 allows a single TCP/IP address space to drive multiple instances of any supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration data set. The LINK statements show how to define a network interface link associated with the device and are included with the DEVICE statement for that device type.

Listed below are the minimum required statements to define a network interface for use by TCP/IP:

- A set of DEVICE and LINK statements for the appropriate device. Depending on the type of device being defined, additional Profile statements and/or VTAM definitions might be required. For more details, see the DEVICE and LINK statements for the device type.
- A HOME statement which assigns an IP address to the LINK interface. For more details, see "HOME Statement" on page 207.
- If you are using static routing, define a GATEWAY statement referencing the LINK interface, to reach the target networks. For more details, see "GATEWAY Statement" on page 193.
- If you are using dynamic routing, see "Chapter 25. Configuring the OROUTED Server" on page 917 and "Chapter 26. Configuring OMPROUTE" on page 949.

Because devices (except for VIPA devices) are not automatically initialized, you must also specify a START statement in the configuration data set to start each device automatically.

Note: The following restrictions apply to specifying DEVICE and LINK statements:

- Because TCP/IP has a maximum of 255 started devices (not including VIPA), you can not start more that 255 devices.
- If you are using OROUTED or OMPROUTE, the maximum number of interfaces that can be specified in the HOME list is 255. This maximum does not include VIPA interfaces.
- There is no maximum for static VIPA interfaces, but the maximum number of Dynamic VIPA interfaces is 64.

There are DEVICE and LINK statements to configure the following:

- ATM devices
- CLAW devices (for example, channel-attached RISC System/6000)
- CTC devices
- HYPERchannel A220 devices
- LCS devices
- MPCIPA devices
- MPCOSA devices
- MPCPTP devices
- SNA LU0 links
- SNA LU6.2 links
- Virtual IP address (VIPA)
- X.25 NPSI connections
- 3745/46 Channel DLC Devices

You can add new DEVICE and LINK statements using the VARY TCPIP,,OBEYFILE command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

For more information on VARY TCPIP, see "VARY Command—TCPIP Address Space" on page 264.

Modifying DEVICE and LINK Statements

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCPIP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

Relationship to VTAM Configuration

CS for OS/390 provides a set of High Performance Data Transfer (HPDT) services that includes MultiPath Channel (MPC), a high-speed channel interface designed for network protocol use (for example, APPN or TCP/IP). Multiple protocols can either share or have exclusive use of a set of channel paths to an attached platform. The term *MPC+* is used to distinguish this multi-protocol version of MPC from earlier versions that were restricted to APPN usage only.

MPC provides the user with the ability to have multiple device paths defined as a single logical connection. The term *MPC group* is used to define a single MPC connection that can contain multiple read and write paths. The number of read and write paths do not have to be equal, but there must be at least one read and write path defined within each MPC group.

MPC groups are defined using the Transport Resource List (TRL), where each defined MPC group becomes an entry (that is, a TRLE) in the TRL table. The user defines the channel paths that are a part of the group in the TRLE. Each TRLE is identified by a *resource_name*. For ATM, the TRLE also has a *port_name* to identify a particular ATM port. For details on defining a TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

Routing Statements

TCP/IP supports static and dynamic routing. You can define static routes in the IP host using the GATEWAY statement in *hlq*.PROFILE.TCPIP. You can also configure OMPROUTE or OROUTED to implement the Routing Information Protocol (RIPv1 and RIPv2) for dynamic routing. Or, you can configure OMPROUTE to implement the Open Shortest Path First (OSPF) protocol for dynamic routing. The use of the GATEWAY statement and BSDROUTINGPARMS statement in PROFILE.TCPIP differs for each of these methods. There are some general rules you can follow when configuring OMPROUTE or OROUTED.

OMPROUTE

- Use the GATEWAY statement in PROFILE.TCPIP to define any static routes that will not be used for dynamic routing. It is highly recommended that static routing not be used to allow routes to be dynamically updated in response to network topology changes.
- If there are adjacent routers that are not running OSPF or RIP, it is recommended that you define static routes in the GATEWAY statement so that OMPROUTE will not modify them.
- The BSDROUTINGPARMS statement in PROFILE.TCPIP is not used. Instead, the interface characteristics are defined in the OMPROUTE configuration file.

  Note: If using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS statement is required to route Transport PDUs prior to OMPROUTE activation. Because the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both the BSDROUTINGPARMS statement and the OMPROUTE configuration file.
- The OMPROUTE server uses the following search order to locate the OMPROUTE configuration data set or file. Only the first file in the search order that can be opened is read to determine the configuration statements.
  1. If the environment variable OMPROUTE_FILE has been defined, OMPROUTE uses this value as the name of an MVS data set or HFS file to access the configuration file.
  2. /etc/omproute.conf
  3. *hlq*.ETC.OMPROUTE.CONF

OROUTED

- Use the GATEWAY statement in PROFILE.TCPIP to define any static routes that will not be used for dynamic routing. It is highly recommended that static routing not be used to allow routes to be dynamically updated in response to network topology changes.
- If there are adjacent routers that are not running RIP, it is recommended that you define passive routes in the OROUTED gateways file so that OROUTED will not modify them. The passive routes are treated as static routes.
- Use BSDROUTINGPARMS in PROFILE.TCPIP to define the interface characteristics for each link that will be used for dynamic routing.
- The OROUTED server uses the following search order to locate the PROFILE configuration data set or file. Only the first file in the search order that can be opened is read to determine the profile statements.
  1. If the environment variable ROUTED_PROFILE has been defined, OROUTED uses this value as the name of an MVS data set or HFS file to

access the profile. The syntax for an MVS data set name is //'*mvs.dataset.name*'. The syntax for an HFS file name is /*directory/subdirectory/file.name*.

2. /etc/routed.profile
3. *hlq*.ROUTED.PROFILE

- The OROUTED server uses the following search order to locate the GATEWAYS configuration data set or file. Only the first file in the search order that can be opened is read to determine the gateway statements.
  1. If the environment variable GATEWAYS_FILE has been defined, OROUTED uses this value as the name of an MVS data set or HFS file to access the gateways file. The syntax for an MVS data set name is //'*mvs.dataset.name*'. The syntax for an HFS file name is /*directory/subdirectory/file.name*.
  2. /etc/gateways
  3. *hlq*.ETC.GATEWAYS

Multiple Network Attachments Support

TCP/IP supports multiple attachments and IP addresses on the same network, providing redundant paths to other hosts or routers on directly-attached networks. When multiple attachments to the same network are configured, one attachment is the primary path to hosts and routers on that network, and others are secondary.

In general, the interface used for the outbound traffic will be the one that contains the first active route to the destination according to the IP routing table. If the destination route is not available, then the interface used will be the one that contains the first active default route.

For outbound traffic originating at the host, the home IP address of the selected interface will be used as the local address of an application socket, if NOSOURCEVIPA is specified and the application did not bind the socket to a specific local IP address.

If using OROUTED and directly-attached multiple network interfaces are defined, the first link for each defined network in the list of HOME addresses will be used as the primary source for routing outbound traffic to the local network as long as the interface is functioning. All other interfaces on a directly-attached network will be used for secondary routing in the event of an interface failure.

If MULTIPATH is configured and there are directly-attached routes across multiple interfaces, either defined in the GATEWAY statement or learned dynamically from OMPROUTE (OSPF/RIP), the interface used for the outbound traffic will be selected on a round-robin basis according to the IP routing table as long as the interface or route is functioning.

Inbound traffic is not affected, because assignment of paths into MVS is done by routing protocols out in the network.

Devices That Support ARP Offload

Certain devices provide an ARP offload function which offloads all ARP processing to the adapter. TCP/IP cannot display any ARP cache information or ARP counter statistics for such devices because the adapter does not provide this data to TCP/IP. The TCP/IP functions affected are:

- NETSTAT ARP onetstat -R

- SNMP ipNetToMediaTable
- SIOCGMONDATA ioctl() ARP Counter statistics

The following devices provide an ARP offload function and do not provide any ARP cache data or ARP counter data to TCP/IP:

- MPCIPA (OSA-Express Gigabit Ethernet)
- MPCOSA (OSA-2 Fast Ethernet, FDDI)

Summary of TCP/IP Configuration Statements

See "Summary of Values (Default, Minimum, and Maximum) for TCP/IP Configuration Statements" on page 120 for a list of TCP/IP configuration statements, their corresponding values, and a table of references for more information.

Sample Profile Configuration Data Set (SAMPPROF)

The following sample configuration data set is provided in *hlq*.SEZAINST(SAMPPROF). This member is used to configure the TCP/IP address space.

```
;
; PROFILE.TCPIP
; =============
;
; This is a sample configuration file for the TCPIP address space
;
; SMP/E distribution name: EZAEB025
;
; COPYRIGHT = NONE
;
; Notes:
;
; - The device configuration, home and routing statements MUST be
;   changed to match your hardware and software configuration.
;   Likewise, the BEGINVTAM section MUST be changed to match your
;   VTAM configuration.
;
; - Lines beginning with semi-colons are comments.  To use a line
;   for your configuration, remove the semi-colon.
;
; - For more information about this file, see the IP Configuration Guide
;
; ====================================================================
; General TCP/IP address space configuration
; ====================================================================
;
; ARPAGE: Specifies the number of minutes between creation or
;   revalidation of an LCS ARP table entry and the deletion of the
;   entry.
;
ARPAGE 20
;
;
; ASSORTEDPARMS: Passes initialization parameters to TCPIP.  However,
;   use of this statement is being phased out.  Use the GLOBALCONFIG,
;   IPCONFIG, TCPCONFIG, and UDPCONFIG statements instead.
;
; GLOBALCONFIG: Provides settings for the entire TCP/IP stack
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
; IPCONFIG: Provides settings for the IP layer of TCP/IP.
```

```
;
; Example IPCONFIG for single stack/single system:
;
IPCONFIG DATAGRAMFWD VARSUBNETTING SYSPLEXROUTING
;
; Example IPCONFIG for automatic activation of inter-stack dynamic XCF
;   and Same Host (IUTSAMEH) links
;
; IPCONFIG DYNAMICXCF 201.1.10.10 255.255.255.0 2
;
; KEEPALIVEOPTIONS: Specifies operating parameters of the TCP keep-alive
;   packets.  These same parameters can be specified on the TCPCONFIG
;   statement as well; use of the TCPCONFIG statement is recommended.
;
;
; SOMAXCONN: Specifies maximum length for the connection request queue
;   created by the socket call listen().
;
SOMAXCONN 10
;
;
; TCPCONFIG: Provides settings for the TCP layer of TCP/IP.
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
;
;
; UDPCONFIG: Provides settings for the UDP layer of TCP/IP
;
UDPCONFIG UNRESTRICTLOWPORTS
;
;
; ====================================================================
; Hardware definitions
; ====================================================================
;
; DEVICE: Defines name (and sometimes device number) for various types
;   of network devices
; LINK: Defines a network interface to be associated with a particular
;   device
;
;
; DEVICE and LINK for CTC devices
;
;   DEVICE  CTC1    CTC  D00  AUTORESTART
;   LINK    CTCD00  CTC  0  CTC1
;
; DEVICE and LINK for HYPERchannel A220 devices:
;
;   DEVICE  HCH1    HCH  E00  AUTORESTART
;   LINK    HCHE00  HCH  1  HCH1
;
; DEVICE and LINK for LAN Channel Station and OSA devices:
;   DEVICE: Defines name and hexadecimal device number for an IBM 8232
;     LAN channel station (LCS) device, and IBM 3172 Interconnect
;     Controller, an IBM 2216 Multiaccess Connector Model 400,
;     an IBM FDDI, Ethernet, or Token Ring OSA, or an IBM ATM OSA-2
;     in LAN emulation mode
;   LINK: Defines a network interface link associated with an LCS
;     device; may be for Ethernet Network, Token-Ring Network or
;     PC Network, or FDDI.
;
;   Example: LCS1 is a 3172 model 1 with a Token Ring and Ethernet
;    adapter
;
;     DEVICE  LCS1  LCS  BA0  AUTORESTART
;     LINK    TR1   IBMTR  0  LCS1
;     LINK    ETH1  ETHERNET  1  LCS1
```

```
;
;   Example: LCS2 is a 3172 model 2 with a FDDI adapter
;
;     DEVICE  LCS2   LCS  BE0  AUTORESTART
;     LINK    FDDI1  FDDI  0  LCS2
;
; DEVICE and LINK for MPCIPA devices:
;
;   DEVICE  MPCIPA1     MPCIPA  NONROUTER AUTORESTART
;   LINK    MPCIPALINK  IPAQGNET  MPCIPA1
;
; DEVICE and LINK for MPCPTP devices:
;
;   DEVICE  MPCPTP1    MPCPTP   AUTORESTART
;   LINK    MPCPTPLINK MPCPTP   MPCPTP1
;
; DEVICE and LINK for CLAW devices:
;
;   DEVICE  RS6K    CLAW  6B2  HOST  PSCA  NONE  26 26 AUTORESTART
;   LINK    IPLINK1 IP  0  RS6K
;
; DEVICE and LINK for SNA LU0 links:
;
;   DEVICE  SNALU0  SNAIUCV   SNALINK  LU000000  SNALINK  AUTORESTART
;   LINK    SNA1    SAMEHOST  1  SNALU0
;
; DEVICE and LINK for SNA LU 6.2 links:
;
;   DEVICE  SNALU621 SNALU62 SNAPROC AUTORESTART
;   LINK    SNA2     SAMEHOST  1  SNALU621
;
; DEVICE and LINK for X.25 NPSI connections:
;
;   DEVICE  X25DEV   X25NPSI  TCPIPX25  AUTORESTART
;   LINK    X25LINK  SAMEHOST  1  X25DEV
;
; DEVICE and LINK for 3745/46 Channel DLC Devices:
;
;   DEVICE  CDLC1    CDLC  C00 AUTORESTART
;   LINK    CDLCLINK CDLC  1  CDLC1
;
; DEVICE and LINK for MPC OSA Fast Ethernet Devices:
;
;   DEVICE  MENET1   MPCOSA   AUTORESTART
;   LINK    ENETLINK OSAENET 0 MENET1
;
; DEVICE and LINK for MPC OSA FDDI Devices:
;
;   DEVICE  MFDDI1   MPCOSA   AUTORESTART
;   LINK    FDDILINK OSAFDDI 0 MFDDI1
;
;
; ------------------------------------------------------------------
; Virtual device definitions
; ------------------------------------------------------------------
;
; DEVICE and LINK for Virtual Devices (VIPA):
;
;   DEVICE  VDEV1   VIRTUAL  0
;   LINK    VLINK1  VIRTUAL  0  VDEV1
;
; Dynamic Virtual Devices can be defined on this system.  This system
; can serve as backup for Dynamic Virtual Devices on other systems.
; A predefined range will allow Dynamic Virtual Devices to be defined
; by IOCTL or Bind requests.
;
; VIPADYNAMIC
```

```
;   Define two dynamic VIPAs on this stack:
;   VIPADEFINE 255.255.255.192 201.2.10.11 201.2.10.12
;
;   Define this stack as backup for these dynamic VIPAs on
;   other TCP/IP stacks:
;   VIPABACKUP 100             201.2.10.13 201.2.10.14
;   VIPABACKUP  80             201.2.10.21 201.2.10.22
;   VIPABACKUP  60             201.2.10.31 201.2.10.33
;   VIPABACKUP  40             201.2.10.32 201.2.10.34
;
;   VIPARANGE DEFINE 255.255.255.192 201.2.10.192
; ENDVIPADYNAMIC
;
; ----------------------------------------------------------------------
; ATM hardware definitions
; ----------------------------------------------------------------------
;
; ATMLIS: Describes characteristics of an ATM logical IP subnet (LIS).
;
; DEVICE and LINK for ATM devices: (See above)
;
; ATMPVC: Describes a permanent virtual circuit (PVC) to be used by an
;   ATM link.
;
; ATMARPSV: Designates the ATMARP server that will resolve ATMARP
;   requests for a logical IP subnet (LIS).
;
; ATMLIS   LIS1   9.67.100.0 255.255.255.0
; DEVICE   OSA1   ATM  PORTNAME  PORT1
; LINK     LINK1  ATM  OSA1  LIS LIS1
; ATMPVC   PVC1   LINK1
; ATMARPSV ARPSV1 LIS1 PVC PVC1
;
;
; ----------------------------------------------------------------------
; Other device statements
; ----------------------------------------------------------------------
;
; START: Starts a device that is currently stopped.
;
; START LCS1
; START LCS2
;
;
; TRANSLATE: Indicates a relationship between an internet address and
;   the network address on a specified link.
;
; TRANSLATE
;   9.67.43.110   FDDI    FF0000006702   FDDI1
;   9.37.84.49    HCH     FF0000005555   HCHE00
;
;
; ======================================================================
; HOME addresses
; ======================================================================
;
; HOME: Provides the list of home IP addresses and associated link names
;
;   - The LOOPBACK statement of 14.0.0.0 should only be used if the
;     installation has applications that require this old loopback
;     address.  The current stack uses 127.0.0.1 as the loopback
;     address.
;
; HOME
;  14.0.0.0       LOOPBACK
;  130.50.75.1    TR1
;  193.5.2.1      ETH1
```

```
;  9.67.43.110    FDDI1
;  193.7.2.1      SNA1
;  9.67.113.80    CTCD00
;  9.37.84.49     HCHE00
;  9.67.113.81    MPCIPALINK
;  9.67.113.82    MPCPTPLINK
;  9.67.114.02    IPLINK1
;  9.67.43.03     SNA2
;  9.67.115.85    X25LINK
;  9.67.116.86    VLINK1
;  9.67.117.87    CDLCLINK
;  9.67.100.80    LINK1
;  9.37.112.13    ENETLINK
;  9.37.112.14    FDDILINK
;
;
; PRIMARYINTERFACE: Specifies which link is designated as the default
;   local host for use by the GETHOSTID() function.
;
;   - If PRIMARYINTERFACE is not specified, then the first link in
;     the HOME statement is the primary interface, as usual.
;
; PRIMARYINTERFACE TR1
;
;
; ========================================================================
; Routing configuration
; ========================================================================
; ------------------------------------------------------------------------
; Static routing
; ------------------------------------------------------------------------
;
; GATEWAY: Defines static routes to the IP route table.
;
; GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
;  Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
;
;  130.50       =         TR1       2000       0.0.255.0    0.0.75.0
;  193.5.2      =         ETH1      1500       0
;  9            =         FDDI1     4000       0.255.255.0  0.67.43.0
;  193.7.2.2    =         SNA1      2000       HOST
;
;
; Indirect Routes - Routes that are reachable through routers on my
;                   network.
;
;  Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
;
;  193.12.2 130.50.75.10  TR1       2000       0
;  10.5.6.4 193.5.2.10    ETH1      1500       HOST
;
;
; Default Route - All packets to an unknown destination are routed
;                 through this route.
;
;  Network  First Hop   Link Name Packet Size  Subnet Mask  Subnet Value
;
;  DEFAULTNET 9.67.43.99  FDDI1    DEFAULTSIZE  0
;
;
; ------------------------------------------------------------------------
; Dynamic routing
; ------------------------------------------------------------------------
;
```

```
; BSDROUTINGPARMS: Defines the characteristics of each link defined at
;   the host over which OROUTED will send routing information to
;   adjacent routers running the RIP protocl and which NCPROUTE will
;   send transport PDUs to client NCPs.
;
;   - OMPROUTE is the recommended routing daemon.  It does not use
;     BSDROUTINGPARMS.
;
;   - OROUTED users must define BSDROUTINGPARMS.
;
;   - Use of the GATEWAY statement (static routes) with the OMPROUTE
;     OROUTED routing daemons is not recommended.
;
; BSDROUTINGPARMS TRUE
;  Link name      MTU    Cost metric  Subnet Mask    Dest address
;   TR1          2000         0       255.255.255.0     0
;   ETH1         1500         0       255.255.255.0     0
;   FDDI1        4000         0       255.255.255.0     0
;   VLINK1    DEFAULTSIZE     0       255.255.255.0     0
;   CTCD00       65527        0       255.255.255.0  9.67.113.90
; ENDBSDROUTINGPARMS
;
;
; ====================================================================
; Application configuration
; ====================================================================
;
; AUTOLOG: Supplies TCPIP with the procedure names to start and the
;  timeout value to use for a hung procedure during AUTOLOG.
;
; AUTOLOG 5
;  FTPD JOBNAME FTPD1   ; FTP Server
;  LPSERVE              ; LPD Server
;  NAMED                ; Domain Name Server
;  NCPROUT              ; NCPROUTE Server
;  OROUTED              ; OROUTED Server
;  OSNMPD               ; SNMP Agent Server
;  PORTMAP              ; Portmap Server
;  RXSERVE              ; Remote Execution Server
;  SMTP                 ; SMTP Server
;  SNMPQE               ; SNMP Client
;  TCPIPX25             ; X25 Server
; ENDAUTOLOG
;
;
; PORT: Reserves a port for specified user IDs, procedures or job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
;
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
    20 TCP OMVS      NOAUTOLOG ; FTP Server
    21 TCP FTPD1               ; FTP Server
    23 TCP INTCLIEN            ; Telnet Server
    25 TCP SMTP                ; SMTP Server
    53 TCP NAMED               ; Domain Name Server
    53 UDP NAMED               ; Domain Name Server
   111 TCP PORTMAP             ; Portmap Server
   111 UDP PORTMAP             ; Portmap Server
```

```
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   162 UDP SNMPQE              ; SNMP Query Engine
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
   515 TCP LPSERVE             ; LPD Server
   520 UDP OROUTED             ; OROUTED Server
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
  3000 TCP CICSTCP             ; CICS Socket
;
;
; PORTRANGE: Reserves a range of ports for specified user IDs,
;   procedures or job names.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;
; SACONFIG: Configures the TCP/IP SNMP subagent
;
SACONFIG ENABLED COMMUNITY public AGENT 161
;
;
; ----------------------------------------------------------------------
; Configure Telnet
; ----------------------------------------------------------------------
;
; TELNETPARMS: Configure the Telnet Server
;
; - TN3270(E) server port 23 options
;
TelnetParms
  Port 23                       ; Port number 23 (std.)
  CodePage ISO8859-1 IBM-1047   ; Linemode ASCII, EBCDIC code pages
  Inactive 0                    ; Let connections stay around
  PrtInactive 0                 ; Let connections stay around
  TimeMark 600
  ScanInterval 120
  SMFinit std
  SMFterm std
  WLMClusterName
    TN3270E
  EndWLMClusterName
EndTelnetParms
;
; TelnetParms
;   Secureport 992 Keyring HFS /tmp/telnet.kdb
; EndTelnetParms
;
; BEGINVTAM: Defines the VTAM parameters required for the Telnet server.
;
BeginVTAM
  Port 23 ; 992
  ; Define logon mode tables to be the defaults shipped with the
  ; latest level of VTAM
  TELNETDEVICE 3278-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3279-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3278-4-E NSX32704 ; 48 line screen -
```

```
                                ; default of NSX32702 is 24
  TELNETDEVICE 3279-4-E NSX32704 ; 48 line screen -
                                ; default of NSX32702 is 24
  TELNETDEVICE 3278-5-E NSX32705 ; 132 column screen-
                                ; default of NSX32702 is 80
  TELNETDEVICE 3279-5-E NSX32705 ; 132 column screen -
                                ; default of NSX32702 is 80

  ; Define the LUs to be used for general users.
  DEFAULTLUS
    TCPABC01..TCPABC99
  ENDDEFAULTLUS

  LUSESSIONPEND     ; On termination of a Telnet server connection,
                    ; the user will revert to the DEFAULTAPPL
                    ; instead of having the connection dropped

  MSG07             ; Sends a USS error message to the client if an
                    ; error occurs during session establishment
                    ; instead of dropping the connection

  DEFAULTAPPL TSO   ; Set the default application for all TN3270(E)
                    ; Telnet sessions to TSO

  LINEMODEAPPL TSO  ; Send all line-mode terminals directly to TSO.

; ALLOWAPPL SAMON QSESSION  ; SAMON appl does CLSDST Pass to next appl

  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
              ; applications.
              ; TSO is multiple applications all beginning with TSO,
              ; so use the * to get them all.  If a session is closed,
              ; disconnect the user rather than log off the user.

  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.

  RESTRICTAPPL IMS ; Only 3 users can use IMS.
    USER USER1     ; Allow user1 access.
      LU TCPIMS01  ; Assign USER1 LU TCPIMS01.
    USER USER2     ; Allow user2 access from the default LU pool.
    USER USER3     ; Allow user3 access from 3 Telnet sessions,
                   ; each with a different reserved LU.
      LU TCPIMS31 LU TCPIMS32 LU TCPIMS33

  ;   Map Telnet sessions from IP address 130.50.10.1 to display the
  ;   USSMSG10 screen from USS table USSAPC.

; USSTCP USSAPC 130.50.10.1

  ;   Map Telnet sessions from the SNA1 link to display the USSMSG10
  ;   screen from USS table USSCBA.

; USSTCP USSCBA SNA1

; LUGROUP  LUGRP1
;       TCPM0001..TCPM0999
;       TCPM1001
; ENDLUGROUP

; LUGROUP LUGRP2
;       TCPM2001  TCPM2003 TCPM2004
;       TCPM0AAA..TCPM0ZZZ
; ENDLUGROUP

  ; Define groups of host names
; HNGROUP HNGRP1
```

```
;      TEST1.TCP.RALEIGH.IBM.COM
;      TEST2.TCP.RALEIGH.IBM.COM
;      *.*.RALEIGH.IBM.COM
; ENDHNGROUP

; HNGROUP HNGRPALL
;      **.COM
; ENDHNGROUP

  ; Map LUs to groups for host names
; LUMAP LUGRP1 HNGRP1
; LUMAP LUGRP2 HNGRPALL
; LUMAP TCPM5000  SPECIAL.TCP.RALEIGH.IBM.COM
EndVTAM
;
;
; ====================================================================
; Diagnostic data statements
; ====================================================================
;
; - For optimum performance, use of tracing should be limited to when
;   required for problem analysis.
;
; ITRACE: Controls TCP/IP run-time tracing
;
; ITRACE ON CONFIG 1
; ITRACE OFF SUBAGENT
;
;
; PKTTRACE: Controls the packet trace facility in TCP/IP.
;
; PKTTRACE ABBREV=200 LINKNAME=TR1 PROT=ICMP IP=*
;   SRCPORT=5000 DESTPORT=161
;
;
; SMFCONFIG: Provides SMF logging for Telnet, FTP, TCP API and TCP
;   stack activity.
;
;   - The SMF record type for TCP/IP records is 118.
;
; SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;
;
; SMFPARMS: Logs the use of TCP by applications using SMF log records.
;   However, use of the SMFCONFIG statement is recommended instead.
;
;
; ====================================================================
; Other statements
; ====================================================================
;
; DELETE: Removes an ATMARPSV, ATMLIS, ATMPVC, device, link, port or
;   portrange.  This statement is typically done via an obey file, not
;   in an initial profile.
;
; STOP: Stops a device.  If used, this statement is typically put in
;   an obey file, not in an initial profile.
;
; INCLUDE: Causes another data set that contains profile configuration
;   statements to be included at this point.
;
;
; --------------------------------------------------------------------
```

Note: In the above example, the numerical value following the VIPABACKUP keyword indicates a *weight* assigned to that stack in terms of backing up the

specified address. The higher the weight, the higher priority of this stack to be used as the primary backup for this address.

Cross-System Coupling Facility (XCF) Dynamics

XCF Dynamics creates dynamic definitions for DEVICE, LINK, HOME, and BSDROUTINGPARMS statements, and for the START command. Dynamic XCF devices and links, when activated, appear to the stack as though they had been defined in the TCP/IP profile. They can be displayed using standard commands, and they can be stopped and started. XCF Dynamics is activated via the DYNAMICXCF keyword on the IPCONFIG statement, which is described in detail below.

XCF Dynamics can be used to generate dynamic definitions for TCP/IP stacks that reside on another OS/390 MVS host in a sysplex and for additional TCP/IP stacks that reside on the same OS/390 MVS host.

The minimum requirements in order for TCP/IP stacks to utilize XCF Dynamics differ based on whether SAMEHOST or inter-host communication is being used. In order to generate definitions for two TCP/IP stacks that reside on different MVS hosts:

- Both MVS hosts must belong to the same sysplex
- VTAM must have XCF communications enabled by specifying XCFINIT=YES as a startup parameter or by activating the VTAM major node, ISTLSXCF. For details about configuration, refer to *OS/390 SecureWay Communications Server: SNA Operation*.
- IPCONFIG DYNAMICXCF must be specified in the TCP/IP profile

With this configuration, both SAMEHOST and inter-host communication can be performed using XCF Dynamics.

In order to generate definitions for two TCP/IP stacks that reside on the same MVS host, you are required to specify IPCONFIG DYNAMICXCF in the TCP/IP profile.

At initialization, each TCP/IP stack configured for XCF (via activation of the VTAM XCF major node, ISTLSXCF) joins a well-known XCF group. For details about configuration, refer to *OS/390 SecureWay Communications Server: SNA Operation*. When other stacks in the group discover the new stack, the definitions are created automatically, the links are activated, and the remote IP address for each link is added to the routing table. After the remote IP address has been added, IP traffic proceeds as usual.

In VTAM, you must activate the XCF major node. You can do this using the start option XCFINIT=YES. If dynamically-defined XCF definitions have been created for another VTAM in the sysplex that has since stopped and restarted with a different CPName, XCF Dynamics recognizes this situation and automatically modifies existing definitions to accommodate the CPName change. If the XCF major node is inactive when TCP/IP is started and the XCF major is not activated until after TCP/IP has finished initialization, TCP/IP will not generate any dynamic definitions for other TCP/IP hosts already started in the sysplex until either:

- A new TCP/IP host is detected
- A profile related operator command is issued (such V TCPIP,,OBEY or a START or STOP command)

To request dynamics for XCF or SAMEHOST connections, enter the following in the IPCONFIG statement:

```
DYNAMICXCF IPAddress SubnetMask CostMetric
```

You can delete dynamically defined XCF devices and links by first stopping the devices to be deleted and then issuing a VARY TCPIP,,OBEY command that contains a DELETE LINK EZAXCFxx and DELETE DEVICE. Because the HOME statement processing does not affect dynamically-defined XCF HOME list entries, the HOME nn.nn.nn.nn EZAXCFxx entry is automatically deleted by DELETE LINK.

Notes:

1. The IP address of the dynamically-defined devices can be changed. Because XCF dynamics uses the same IP address for all of the dynamically-defined devices, all of the dynamic devices IP addresses will be changed. An individual dynamic device cannot be changed. To change the IP addresses:
   a. Stop all of the dynamically-defined devices.
   b. Issue the VARY TCPIP,,OBEY command, which contains the changed IP address on the DYNAMICXCF statement.

   After they have all stopped, XCF Dynamics will change the IP address and automatically restart all of the dynamically-defined devices. XCF Dynamics changes the IP address for dynamically-defined XCF, SAMEHOST links, or both in exactly the same way (with the same operational characteristics) as if you had changed the IP address for static XCF or SAMEHOST definitions and then executed VARY OBEY.
2. Because the interfaces generated by XCF Dynamics use a single IP address, the output of the SIOCGIFCONF ioctl() contains multiple entries with the same IP address. If an application is using the SIOCGIFCONF output to issue bind() to all the entries returned, the application could receive EADDRINUSE on a bind() if there are multiple XCF devices defined by XCF Dynamics in the list.
3. If GATEWAY statements are being used to define your routing, any GATEWAY statement that refers to a dynamically-defined device must be issued by a VARY TCPIP,,OBEY command after the dynamic devices have been defined and started. This GATEWAY statement must be in a VARY TCPIP,,OBEY data set separate from the data set used to define the dynamic devices (either initial Profile data set or another VARY TCPIP,,OBEY data set).
4. Even though the HOME, BSDROUTINGPARMS, and GATEWAY definitions are full replacement keywords, the definitions generated by XCF Dynamics will not replace any existing definitions. Likewise, user-defined HOME, BSDROUTINGPARMS, and GATEWAY definitions will not affect existing or future definitions generated by XCF Dynamics.

If the received CPName is the same as that for the receiving stack, if no device exists with the name IUTSAMEH, and if no link exists with the name EZASAMEMVS, internal definitions equivalent to the following are created:

DEVICE IUTSAMEH MPCPTP AUTORESTART
Device definition to obtain the most efficient stack-to-stack communications within the same MVS image.

LINK EZASAMEMVS MPCPTP IUTSAMEH
Link definition for the IUTSAMEH device.

HOME IPAddress EZASAMEMVS
Associates the IP address with the SAMEHOST link.

START IUTSAMEH Starts the SAMEHOST device.

BSDROUTINGPARMS EZASAMEMVS DEFAULTSIZE CostMetric SubnetMask DestIPAddress Defines a new link to the OROUTED routing daemon.

Note: The DestIPAddress is always zero.

If the received CPName is different than the one for the current stack, if no device with that name exists, and if no link exists with the name EZAXCFxx (where xx is the value of the MVS system symbol (SYSCLONE) for the MVS hosting the VTAM with the device name), internal definitions equivalent to the following are created:

DEVICE MPCPTP AUTORESTART Device definition to communicate with TCP/IP stacks hosted by the remote VTAM.

LINK EZAXCFnn MPCPTP Link definition for the device, where nn is the SYSCLONE value for the remote VTAM and MVS.

HOME IPAddress EZAXCFnn Associates the IP address with the dynamic XCF link.

START Starts the specified device.

BSDROUTINGPARMS EZAXCFnn DEFAULTSIZE CostMetric SubnetMask DestIPAddress Defines the new link to the OROUTED routing daemon.

Notes:

1. If EZAXCFnn is already defined as a link name or the CPName is already defined as a device name, then Dynamic XCF definitions will not be generated for discovery of another stack in the same MVS image.
2. The DestIPAddress is always zero.

For details about these XCF-related statements, refer to *OS/390 SecureWay Communications Server: IP Configuration*. For information about changes to NETSTAT displays of XCF Dynamics settings, refer to *OS/390 SecureWay Communications Server: IP Configuration*.

Examples of Definitions Generated by XCF Dynamics

Example 1:

This configuration consists of two MVS systems (MVS1,MVS2) that are members of the same sysplex. Each MVS host has one TCP/IP stack (TCPIP1 and TCPIP2, respectively). From the syntax descriptions described above, the following information is needed to generate the dynamic definitions:

- MVS sysclone value
- VTAM CPNAME
- Status of XCF in VTAM
- The values specified on the IPCONFIG DYNAMICXCF keyword

Using the following user definitions:

```
MVS1:
Sysclone = A1
VTAM Cpname = VTAM1
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP1: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.1 255.255.255.248 3
```

```
MVS2:
Sysclone = B2 VTAM
Cpname = VTAM2 VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP2: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.2.2.2 255.255.255.248 2
```

After both TCPIP1 and TCPIP2 have been started, the following definitions will be generated.

TCPIP1 will generate:

```
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 VTAM2
HOME 9.1.1.1 EZAXCFB2
BSDROUTINGPARMS EZAXCFB2 DEFAULTSIZE 3 255.255.255.248 0
START VTAM2
```

TCPIP2 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 VTAM1
HOME 9.2.2.2 EZAXCFA1
BSDROUTINGPARMS EZAXCFA1 DEFAULTSIZE 2 255.255.255.248 0
START VTAM1
```

Example 2:

The configuration is the same as Example 1 except a second TCP/IP stack (TCPIP3) was added to MVS1.

Using the following user definitions:

```
MVS1:
Sysclone = A1
VTAM Cpname = VTAM1
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is activeTCPIP1: PROFILE.TCPIP co
TCPIP3: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.3 255.255.255.248 0
MVS2:
Sysclone = B2
VTAM Cpname = VTAM2
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is activeTCPIP2: PROFILE.TCPIP co
```

After both TCPIP1 and TCPIP2 have been started, the following definitions will be generated, as in Example 1.

TCPIP1 will generate:

```
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 VTAM2
HOME 9.1.1.1 EZAXCFB2
BSDROUTINGPARMS EZAXCFB2 DEFAULTSIZE 3 255.255.255.248 0
START VTAM2
```

TCPIP2 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 VTAM1
HOME 9.2.2.2 EZAXCFA1
BSDROUTINGPARMS EZAXCFA1 DEFAULTSIZE 2 255.255.255.248 0
START VTAM1
```

Now, TCPIP3 is started. TCPIP1 and TCPIP2 recognize that TCPIP3 has started. TCPIP3 will generate definitions for both TCPIP1 and TCPIP2. TCPIP1 will generate SAMEHOST definitions for TCPIP1. However, TCPIP2 does not need to generate and will not generate any new definitions except for routing information for

TCPIP3. New definitions do not need to be created because the DEVICE and LINK definitions are based on the discovery of a new VTAM node in the sysplex. (The DEVICE name uses the VTAM cpname.) TCPIP1 will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS IUTSAMEH
HOME 9.1.1.1 EZASAMEMVS
BSDROUTINGPARMS EZASAMVEMVS DEFAULTSIZE 3 255.255.255.248 0
START IUTSAMEH
```

TCPIP2 does not generate anything.

TCPIP3 will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS IUTSAMEH
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 VTAM2
HOME 9.1.1.3 EZAXCFB2
   9.1.1.3 EZASAMEMVS
BSDROUTINGPARMS EZAXCFB2 DEFAULTSIZE 0 255.255.255.248 0
BSDROUTINGPARMS EZASAMVEMVS DEFAULTSIZE 0 255.255.255.248 0
START IUTSAMEH
START VTAM2
```

Example 3:

To continue Example 2, add another MVS host (MVS3) with a VTAM node (VTAM3)with one TCP/IP stack (TCPIP4).

```
MVS3:
Sysclone = C3
VTAM Cpname = VTAM3
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP4: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.3.3.3 255.255.255.248 0
```

In this example, the previously active TCP/IP stacks will generate definitions for TCPIP4 because a new VTAM stack has become active in the sysplex. TCPIP4 will generate definitions for definitions for TCPIP1/TCPIP3 and TCPIP2.

TCPIP1 will generate:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 VTAM3
HOME 9.1.1.1 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 DEFAULTSIZE 3 255.255.255.248 0
START VTAM3
```

TCPIP2 will generate:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 VTAM3
HOME 9.2.2.2 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 DEFAULTSIZE 2 255.255.255.248 0
START VTAM3
```

TCPIP3 will generate:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 VTAM3
HOME 9.1.1.3 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 DEFAULTSIZE 0 255.255.255.248 0
START VTAM3
```

TCPIP4 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 VTAM1
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2
VTAM2 HOME 9.3.3.3
EZAXCFA1
     9.3.3.3 EZAXCFB2
BSDROUTINGPARMS EZAXCFA1 DEFAULTSIZE 0 255.255.255.248 0
BSDROUTINGPARMS EZAXCFB2 DEFAULTSIZE 0 255.255.255.248 0
START VTAM1 START VTAM2
```

Example 4:

This example will illustrate how XCF dynamics can generate SAMEHOST definitions without VTAM having its XCF enabled.

```
MVS1:
Sysclone = A1  (not used in this example)
VTAM Cpname = VTAM1  (not used in this example)
VTAM has XCFINIT=NO specified and has not activated the major node ISTLSXCF.
TCPIP1: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.1 255.255.255.248 3
TCPIP3: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.3 255.255.255.248 0
```

TCPIP1 will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS IUTSAMEH
HOME 9.1.1.1 EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS DEFAULTSIZE 3 255.255.255.248 0
START EZASAMEMVS
```

TCPIP3 will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS IUTSAMEH
HOME 9.1.1.3 EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS DEFAULTSIZE 0 255.255.255.248 0
START EZASAMEMVS
```

PROFILE.TCPIP Configuration Statements

This section contains the configuration statements for the *hlq*.PROFILE.TCPIP configuration data set and a summary of TCP/IP configuration statements.

For a description of the TELNETPARMS and BEGINVTAM statements, see “Chapter 12. Configuring the Telnet Server” on page 395.

For a list of unsupported statements, see the *OS/390 SecureWay Communications Server: IP Migration*.

Notes:

1. Read the Usage Notes for each profile statement you include in your PROFILE.TCPIP; these notes contain important information about configuring each statement.
2. Read the Modifying sections for instructions about modifying your profile statements.

Summary of Values (Default, Minimum, and Maximum) for TCP/IP Configuration Statements

Table 8 lists the TCP/IP configuration statements and the minimum, maximum, and default parameter values. Table 9 on page 124 contains a brief description of each configuration statement, along with the page where further information can be found. Complete descriptions of the configuration statements follow the tables.

Table 8. Summary of Min, Max, and Default Values for TCP/IP Configuration Statements

STATEMENT **parameter**	**Minimum**	**Maximum**	**Default**
ARPAGE (in minutes)	1	1440	20
ATMLIS			
default_mtu	0	65535	9180
inactivity_timeout	0	65535	300
min_holding_time	0	65535	60
cache_entry_age	60	900	900
arp_retries	0	10	2
arp_timeout	1	60	3
peak_cell_rate	0	2147483647	0
BSDROUTINGPARMS			
mtu (size in bytes)	1	65535	DEFAULTSIZE (576)
cost_metric	0	14	0
DEVICE/LINK ATM			
ifspeed (in bits per second)	0	2147483647	0 (set dynamically)
ifhspeed (million bits per second)	0	2147	0
DEVICE/LINK CTC			
iobuffersize[1]	32K	65535	32K
ifspeed (in bits per second)	0	2147483647	4500000
ifhspeed (million bits per second)	0	2147	4
DEVICE/LINK HYPERCHANNEL A220			
ifspeed (in bits per second)	0	2147483647	50000000
ifhspeed (million bits per second)	0	2147	50
DEVICE/LINK LCS			

Table 8. Summary of Min, Max, and Default Values for TCP/IP Configuration Statements (continued)

STATEMENT **parameter**	**Minimum**	**Maximum**	**Default**
iobuffersize[6]	20480	32768	20480
ifspeed (in bits per second)	0	2147483647	4000000
ifhspeed (million bits per second)	0	2147	4
DEVICE/LINK MPCIPA			
ifspeed (in bits per second)	0	2147483647	1000000000
ifhspeed (million bits per second)	0	2147	1000
DEVICE/LINK MPCPTP			
ifspeed (in bits per second)	0	2147483647	4500000
ifhspeed (million bits per second)	0	2147	4
DEVICE/LINK CLAW			
read_buffers	1	512	15
write_buffers	1	512	15
read_size[2]	1024	4096	4096
write_size[2]	1024	4096	4096
ifspeed (in bits per second)	0	2147483647	100000000
ifhspeed (million bits per second)	0	2147	100
DEVICE/LINK SNA LU0			
ifspeed (in bits per second)	0	2147483647	56000
ifhspeed (million bits per second)	0	2147	0
DEVICE/LINK SNA LU 6.2			
ifspeed (in bits per second)	0	2147483647	56000
ifhspeed (million bits per second)	0	2147	0
DEVICE/LINK X.25 NPSI			
ifspeed (in bits per second)	0	2147483647	56000
ifhspeed (million bits per second)	0	2147	0

Table 8. Summary of Min, Max, and Default Values for TCP/IP Configuration Statements (continued)

STATEMENT **parameter**	**Minimum**	**Maximum**	**Default**
DEVICE/LINK 3745/46 Channel DLC			
read_buffers	1	63	15
write_buffers	1	63	15
read_size[2]	1024	4096	4096
write_size[2]	1024	4096	4096
ifspeed (in bits per second)	0	2147483647	4500000
ifhspeed (million bits per second)	0	2147	4
GATEWAY			
max_packet_size (in bytes)	1	65535	DEFAULTSIZE (576)
maximumretransmittime (in seconds)	0	999.990	120
minimumretransmittime (in seconds)	0	99.990	0.5
roundtripgain	0	1.0	0.125
variancegain	0	1.0	0.25
variancemultiplier	0	99.990	2
IPCONFIG			
arpto (in seconds)	60	86400	1200
cost_metric	0	14	none
reassemblytimeout (in seconds)	1	240	60
ttl	1	255	64
KEEPALIVEOPTIONS			
interval[3] (in minutes)	0	35,791	120
PKTTRACE			
abbrev	1	65535	200
prot	0	255	
port_num	1	65535	
PORT			
port_num	1	65535	
PORTRANGE			

Table 8. Summary of Min, Max, and Default Values for TCP/IP Configuration Statements (continued)

STATEMENT **parameter**	**Minimum**	**Maximum**	**Default**
1st_port	1	65535	
num_ports	1	65535	
SACONFIG			
osasf_port_number	0	65535	
agent_port_number	1	65535	161
SOMAXCONN			
maximum_queue_depth	1	2147483647	10
TCPCONFIG			
default_keepalive_interval (in minutes)	0	35791	120
tcp_send_buffer_size	256	256K	16384 (16K)[4]
tcp_receive_buffer_size	256	tcp_max_receive_ buffer_size	16384 (16K)[4]
tcp_max_receive_buffer_size	tcp_receive_ buffer_size	512K	256K
UDPCONFIG			
udp_send_buffer_size	1	65507	65507
udp_receive_buffer_size	1	65507	65507
VIPABACKUP			
rank	1	254	1

Notes:

1. The only valid values are 32K, 32768, and 65535.
2. These values can only be 1024, 2048, or 4096. For buffer size of 4096, the maximum number of buffers is 15; for 2048, 31; for 1024, 63.
3. KEEPALIVEOPTIONS interval allows zero (0) to turn it off. The maximum value of 35791 is approximately 2**31-1 milliseconds.
4. The default for TCPSENDBFRSIZE and TCPRCVBUFRSIZE is approximately 16384 (16K), but will change slightly with service changes.
5. Read_buffers multiplied by read_size must be less than 64000.
6. The only valid values are 20K, 20480, 32K, and 32768.

Table 9 on page 124 contains a brief description of each configuration statement, along with the page number where you can find more information.

Table 9. Summary of TCP/IP Address Space Configuration Statements

Statement	Description	Page
ARPAGE	Alters the number of minutes before an ARP table entry is deleted.	127
ASSORTEDPARMS	Passes initialization parameters to TCPIP.	128
ATMARPSV	Defines the ATMARP server to resolve arp requests for a logical IP subnetwork (LIS).	131
ATMLIS	Describes the characteristics of an ATM logical IP subnet (LIS).	133
ATMPVC	Describes a permanent virtual circuit to be used by an ATM link.	136
AUTOLOG	Indicates which procedures should be automatically started when TCP/IP is started.	137
BEGINVTAM, ENDVTAM and associated statements	Descriptions of the Telnet statements for VTAMin the *hlq*.PROFILE.TCPIP data set.	429
BSDROUTINGPARMS	Defines network interface information. Used by the OROUTED and NCPROUTE servers.	140
DELETE	Removes an ATMARPSV, ATMLIS, ATMPVC, device, link, port, or portrange.	144
DEVICE and LINK	ATM Devices	148
DEVICE and LINK	CTC devices	155
DEVICE and LINK	HYPERchannel A220 devices	158
DEVICE and LINK	LCS devices	161
DEVICE and LINK	MPC OSA devices	172
DEVICE and LINK	MPCIPA devices	169
DEVICE and LINK	MPCPTP devices	175
DEVICE and LINK	CLAW devices	151
DEVICE and LINK	SNA LU0 links	179
DEVICE and LINK	SNA LU6.2 links	182
DEVICE and LINK	X.25 NPSI connections	185
DEVICE and LINK	Virtual Devices	188
DEVICE and LINK	3745/46 Channel DLC Devices	190
GATEWAY	Defines IP routing table entries for static routes.	193
GLOBALCONFIG	Passes global configuration parameters to TCP/IP.	205
HOME	Provides a list of home addresses and associated link names.	207
INCLUDE	Causes another data set that contains profile configuration statements to be included at this point.	211
INTERNALCLIENTPARMS	See TELNETPARMS.	404

Table 9. Summary of TCP/IP Address Space Configuration Statements (continued)

Statement	Description	Page
IPCONFIG	Specifies IP configuration values.	213
ITRACE	Controls tracing for configuration, the SNMP subagent, commands, and the autolog subtask.	220
KEEPALIVEOPTIONS	Specifies the operating parameters of the TCP keep-alive mechanism.	222
LINK	Defines network interface links.	101
PKTTRACE	Defines the conditions used to select IP packets as candidates for tracing and subsequent analysis.	223
PORT	Reserves a port for one or more given process names.	229
PORTRANGE	Reserves a range of ports for one or more process names.	232
PRIMARYINTERFACE	Specifies which link is to be considered the primary interface.	235
SACONFIG	Specifies parameters for the TCP/IP SNMP subagent.	236
SMFCONFIG	Provides SMF logging for Telnet, FTP, TCP API, and TCP stack activity.	239
SMFPARMS	Provides SMF logging for Telnet and FTP client activity and TCP API activity.	241
SOMAXCONN	Specifies a maximum connection length for the connection request queues created by the socket call listen().	243
START	Starts the specified device.	244
STOP	Stops the specified device.	246
TCPCONFIG	Specifies TCP parameters.	247
TELNETPARMS	Defines parameters for the Telnet server.	404
TRANSLATE	Indicates the relationship between an internet address and the network address.	249
UDPCONFIG	Specifies UDP parameters.	252
VIPADYNAMIC	Starts a block of definitions related to Dynamic VIPAs.	254

Statement Syntax

Statement syntax is the same in both the configuration data set *hlq*.PROFILE.TCPIP and the VARY TCPIP,,CMD=OBEYFILE data set. The following formatting restrictions apply to configuration statements:

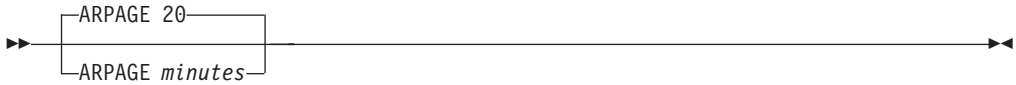
- Entries in a configuration data set are free format; blanks, comments, and end-of-record are ignored.

- A configuration statement consists of a statement name followed by a required blank, and usually one or more positional arguments. Separate each argument by one or more blanks or end-of-record.
- An argument followed by a comment must have a blank before the semicolon.
- A semicolon begins a comment. Comments act as blanks, separating words without affecting their meaning.
- Statements can be split across multiple lines.
- Sequence numbers are not allowed.
- Lowercase letters are translated to uppercase before the statements are executed, except for the community name on the SACONFIG statement. The community name is case sensitive.
- An END*statement* terminates a number of statements, such as ASSORTEDPARMS. If the END*statement* is omitted, all subsequent tokens in the data set are interpreted as parameters for that configuration statement.
- If a syntax error is encountered in a list of parameters, such as a HOME list, the rest of the entries in the list are ignored.

ARPAGE Statement

Use the ARPAGE statement to change the number of minutes between creation or revalidation of an ARP table entry, and deletion of the entry. By default, TCP/IP deletes ARP table entries 20 minutes after creation or revalidation. An ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address. The ARPAGE statement only applies to LAN channel station (LCS) devices.

Syntax



Parameters

minutes
The number of minutes between creation or revalidation of an ARP table entry and deletion of the entry.

This number is an integer in the range of 1 – 1440 (24 hours). The default is 20 minutes.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example clears the ARP tables every 10 minutes.

```
ARPAGE 10
```

Usage Notes

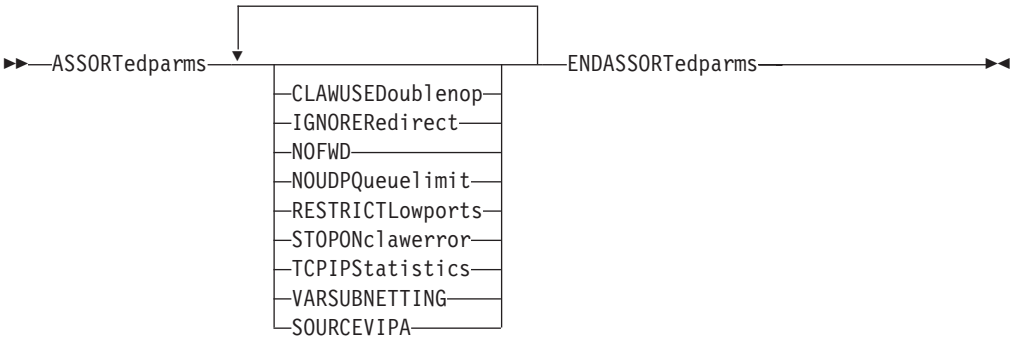
- IPCONFIG ARPTO allows you to specify the number of *seconds* between creation or revalidation and deletion.
- The revalidation of ARP requests for ATM is controlled via the ATMLIS statement.
- Because ARP cache entries for MPCIPA and MPCOSA interfaces are not managed by the TCP/IP stack, they are not affected by the ARPAGE statement.

ASSORTEDPARMS Statement

Use the ASSORTEDPARMS statement to pass initialization parameters to TCP/IP.

Note: Support for the ASSORTEDPARMS statement will be dropped in a future release. It is recommended that you use the GLOBALCONFIG, IPCONFIG, TCPCONFIG, and UDPCONFIG statements instead of ASSORTEDPARMS.

Syntax



Parameters

CLAWUSEDOUBLENOP

Forces channel programs for Common Link Access to Workstation (CLAW) devices to have two NOP CCWs to end the channel programs. This is required for some vendor devices, and only applies to first-level MVS systems. The CLAWUSEDOUBLENOP parameter is confirmed by the message:

```
CLAWUSEDOUBLENOP is set
```

IGNOREREDIRECT

Causes TCP/IP to ignore ICMP Redirect packets. The IGNOREREDIRECT parameter is confirmed by the message:

```
ICMP will ignore redirects
```

If you are using OROUTED, use this option because OROUTED does not support ICMP redirects.

NOFWD

Stops the transfer of data between networks by disabling IP datagram routing between different network interfaces. This statement can be used for security or to ensure correct usage of limited resources. The NOFWD parameter is confirmed by the message:

```
IP forwarding is disabled
```

If either ASSORTEDPARMS NOFWD or IPCONFIG NODATAGRAMFWD is specified in a profile, or if neither the ASSORTEDPARMS nor the IPCONFIG statement is specified, forwarding is disabled. If the ASSORTEDPARMS or IPCONFIG statement is specified and the NOFWD and NODATAGRAMFWD parameters are not included, forwarding is enabled.

NOUDPQUEUELIMIT

Causes TCP/IP to relax the default limit of 21 incoming datagrams queued on a UDP port.

The NOUDPQUEUELIMIT parameter is confirmed by the message:

```
No limit on incoming UDP datagram queue set
```

RESTRICTLOWPORTS

When set, ports 1 through 1023 are reserved for jobs by the PORT and PORTRANGE statement. The RESTRICTLOWPORTS parameter is confirmed by the messages:

```
UDP ports 1 thru 1023 are reserved
TCP ports 1 thru 1023 are reserved
```

Note: When RESTRICTLOWPORTS is specified, an application cannot obtain a port in the 0 through 1023 range unless it is explicitly reserved for that application, APF authorized, or it is not using the Pascal API and has an OMVS UID of zero. Therefore, be careful in specifying this option. Some applications like RSH, LPR, NPF, and others, have a dependency on being able to obtain an available port in the 0 through 1023 range without having that port explicitly reserved for its use. **Use of this parameter is not recommended.**

STOPONCLAWERROR

Stops channel programs (HALTIO and HALTSIO) when a CLAW device error is detected. The STOPONCLAWERROR parameter is confirmed by the message:

```
STOPONCLAWERROR is enabled
```

TCPIPSTATISTICS

Prints the values of several TCP/IP counters to the output data set designated by the CFGPRINT JCL statement. These counters include the number of TCP retransmissions and the total number of TCP segments sent from the MVS TCP/IP system.

The TCPIPSTATISTICS parameter is confirmed by the message:

```
ASSORTEDPARMS TCPIPSTATISTICS is enabled
```

Note: This parameter is identical to the GLOBALCONFIG TCPIPSTATISTICS parameter (see “GLOBALCONFIG Statement” on page 205). Both parameters print identical results in the Global Configuration Information section.

However, the SMFCONFIG TCPIPSTATISTICS parameter (see “SMFCONFIG Statement” on page 239) serves a different purpose. Requests that SMF records of subtype 5 containing TCP/IP statistics are created. Note that these records are created periodically based on the SMF interval in effect.

VARSUBNETTING

Specifies that variable subnetting is being used. Enables variable subnetting and supernetting support in TCP/IP. Variable-length subnet masks may be coded on the GATEWAY and BSDROUTINGPARMS statements. Also, this option allows OMPROUTE and OROUTED applications to dynamically update the IP routing table with variable-length subnet masks. If OROUTED is configured to use RIPv2, VARSUBNETTING must be enabled. If OMPROUTE is started, VARSUBNETTING will be enabled automatically. The VARSUBNETTING parameter is confirmed by the message:

```
Variable Subnetting Support is enabled
```

SOURCEVIPA

Requests TCP/IP to use the corresponding virtual IP address in the HOME list as the source IP address for outbound datagrams. This parameter has no effect

on RIP packets used by RIP services (OROUTED, NCPROUTE, and OMPROUTE) as well as OSPF packets by OSPF services (OMPROUTE). The SOURCEVIPA parameter is confirmed by the message:

```
SOURCEVIPA Support is enabled
```

Modifying

To change a parameter value, you must respecify the statement with the new parameter value. Any parameters not specified will be reset to their default value.

Examples

This example shows the use of the NOFWD parameter on the ASSORTEDPARMS statement.

```
ASSORTEDPARMS
  NOFWD
ENDASSORTEDPARMS
```

Usage Notes

- It is recommended that you use the IPCONFIG, UDPCONFIG, GLOBALCONFIG, and TCPCONFIG statements rather than the ASSORTEDPARMS statement. If some but not all of the ASSORTEDPARMS are specified, by default, those not specified are set to off. IPCONFIG, UDPCONFIG, and TCPCONFIG eliminate this side effect.
- The ENDASSORTEDPARMS statement is the delimiter for the ASSORTEDPARMS statement. If ENDASSORTEDPARMS is omitted, subsequent tokens are saved until the maximum string length is reached, and an error message is generated. The ASSORTEDPARMS string is then cleared, and processing resumes at the next token that is recognized as a valid configuration command.
- If you specify NOUDPQUEUELIMIT when you are running untested applications on your system, a malfunctioning application can use available storage.
- If any of the ASSORTEDPARMS are specified on other statements (IPCONFIG, TCPCONFIG, or UDPCONFIG), the settings from the last statement processed are used. For example, if RESTRICTLOWPORTS is not specified on ASSORTEDPARMS (and thus defaults to off) but is specified on a subsequent TCPCONFIG statement, RESTRICTLOWPORTS will be set for TCP.

Related Topics

- “IPCONFIG Statement” on page 213
- “TCPCONFIG Statement” on page 247
- “UDPCONFIG Statement” on page 252
- “GLOBALCONFIG Statement” on page 205

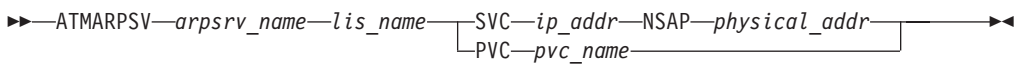
ATMARPSV Statement

Use the ATMARPSV statement to designate the ATMARP server that will resolve ATMARP requests for a logical IP subnet (LIS).

When an ATM device is started, TCP/IP will attempt to establish a connection to the ATMARP server for any LINK associated with a device that both specifies an ATMLIS and has a corresponding ATMARPSV defined.

Some ATMARP server products do not support being used as an ATMARP server over a PVC connection.

Syntax



Parameters

arpsrv_name
The ATMARP server to resolve ARP requests for this LIS. An arpsrv_name has a maximum length of 16 characters.

lis_name
The logical IP subnet (LIS) as defined previously on the ATMLIS statement and as included on the LINK statement. A lis_name has a maximum length of 16 characters.

SVC
Indicates that TCP/IP should connect to the ATMARP server via a switched virtual circuit (SVC).

PVC
Indicates that TCP/IP should connect to the ATMARP server via a permanent virtual circuit (PVC).

ip_addr
The IP address of the ATMARP server. This IP address must be contained within the subnet defined by the *lis_name* parameter.

pvc_name
The PVC name of the connection to the ATMARP server.

NSAP
The type of physical address; Network Services Access Point.

physical_addr
The physical address of the ATMARP server. Specify a 40 digit hexadecimal value.

Modifying

To dynamically change any values on the ATMARPSV statement, follow these steps:

1. Stop the associated ATM device or devices.
2. Use VARY TCPIP with an obeyfile which contains a DELETE ATMARPSV statement.

3. Use VARY TCPIP with an obeyfile which contains the updated ATMARPSV statement.
4. Start the associated ATM device or devices.

Examples

This is an example of a PVC connection to an ATMARP server:

```
ATMLIS   LIS1   9.67.100.0 255.255.255.0
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1  LIS LIS1
ATMPVC   PVC1   LINK1
ATMARPSV ARPSV1 LIS1 PVC PVC1
```

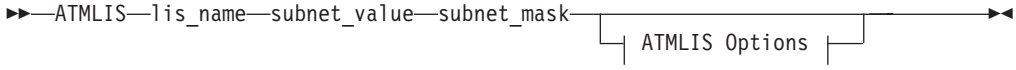
This is an example of a SVC connection to an ATMARP server:

```
ATMLIS   LIS1   9.67.100.0 255.255.255.0
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1  LIS LIS1
ATMARPSV ARPSV1 LIS1 SVC 9.67.100.10
NSAP 12345678901234567890123456789012345678901234567890
```

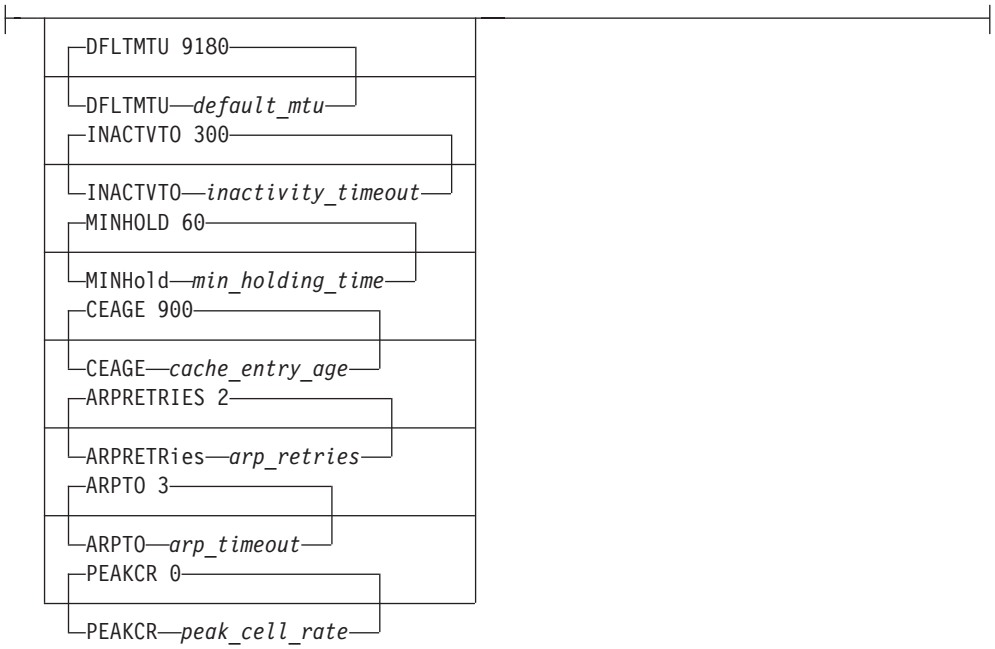
ATMLIS Statement

Use the ATMLIS statement to describe the characteristics of an ATM logical IP subnet (LIS). A LIS is a separate administrative ATM entity. Each logical IP subnet operates and communicates independently of other logical IP subnets on the same ATM network.

Syntax



ATMLIS Options:



Parameters

lis_name
The ATM logical IP subnet on the LINK statement. A lis_name has a maximum length of 16 characters.

subnet_value
The subnet value that defines this logical IP subnet.

subnet_mask
The subnet mask that defines this logical IP subnet.

default_mtu
The maximum transmission unit for SVCs within this logical IP subnet. The minimum valid value for this parameter is 0, the maximum is 65535, and the default is 9180.

inactivity_timeout
The number of seconds before an established SVC connection is dropped due

to no traffic. A value of 0 for this parameter indicates there is no time out period. If a value of 1 through 9 is specified, a value of 10 is used. The maximum value is 65535, and the default is 300.

min_holding_time
The number of seconds that a call remains open. A value of 0 for this parameter indicates that the call will be controlled completely by the inactivity_timeout. The maximum value for this parameter is 65535 and the default is 60.

Note: If min_holding_time is less than inactivity_timeout or if inactivity_timeout is 0, then the value for min_holding_time has no effect.

cache_entry_age
The number of seconds before an ARP cache entry is removed from the cache. The minimum value for this parameter is 60. The maximum and default value is 900.

arp_retries
The number of times an ATMARP request is retried when no response is received and the arp_timeout expires. By default, two retries occur. The minimum value for this parameter is 0 and the maximum is 10.

arp_timeout
The number of seconds to wait before retransmitting an ATMARP request. By default, the wait is 3 seconds. The minimum value for this variable is 1 second and the maximum is 60 seconds.

peak_cell_rate
Indicates the best effort peak cell rate for both forward and backward traffic. A value of 0 indicates that a peak cell rate equal to 10% of the actual link speed is used. This is the default. The maximum value for this variable is 2147483647.

Modifying

The lis_name, subnet_value, and subnet_mask are used to identify each ATMLIS. ATMLIS options can be updated by issuing an ATMLIS statement for an existing ATMLIS with an identical lis_name, subnet_value, and subnet_mask. If a previously defined lis_name is used on another ATMLIS statement with a different subnet_mask and subnet_value, then an error message is issued saying that the ATMLIS is already defined.

To dynamically change any options (other than subnet value and subnet mask) on the ATMLIS statement, use VARY TCPIP with an obeyfile which contains the updated ATMLIS statement. Any options not included on the ATMLIS statement are reset to defaults.

Note: The new ATMLIS values will not apply to any open ATM SVCs, but they will apply to any newly created ATM SVCs.

To dynamically change the subnet value or subnet mask value on the ATMLIS statement, follow these steps:

1. Stop the associated ATM devices or devices.
2. Use VARY TCPIP with an obeyfile which contains a DELETE ATMLIS statement and a DELETE LINK statement for each associated ATM link and a DELETE ATMARPSV statement for any associated ATMARPSV.
3. Use VARY TCPIP with an obeyfile which contains the updated ATMLIS statement along with the associated ATM LINK and ATMARPSV statements.

4. Start the associated ATM device or devices.

Examples

```
ATMLIS   LIS1   9.67.100.0  255.255.255.0
```

Usage Notes

- The subnet_value must be in the subnet_mask.
- The subnet_value must be a class A, B, or C address.
- An ATMLIS must be referenced by a LINK statement. If an ATMLIS is unreferenced by any LINK statement, that ATMLIS and any ATMARPSV referring to that ATMLIS are automatically deleted.
- A HOME address used by an ATM LINK referencing an ATMLIS should be within the logical IP subnetwork defined by the LIS subnet_value and subnet_mask. If it is not within the subnetwork the LINK will not be able to send or receive data over SVCs.

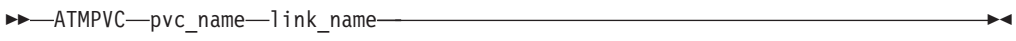
Related Topics

- “ATMARPSV Statement” on page 131
- “DEVICE and LINK Statement—ATM Devices” on page 148

ATMPVC Statement

Use the ATMPVC statement to describe a permanent virtual circuit (PVC) to be used by an ATM link.

Syntax



Parameters

pvc_name
The permanent virtual circuit on the ATM network. A pvc_name has a maximum length of 8 characters.

link_name
The name of the ATM link associated with this PVC. The link_name must be defined previously with a LINK statement.

Modifying

To dynamically change any values on the ATMPVC statement, follow these steps:

1. Stop the associated ATM device whose link is referenced on the ATMPVC statement.
2. Use VARY TCPIP with an obeyfile which contains a DELETE ATMPVC statement.
3. Use VARY TCPIP with an obeyfile which contains the updated ATMPVC statement.
4. Start the associated ATM device.

Examples

```
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1
ATMPVC   PVC1   LINK1
```

Usage Notes

The link_name must be defined via a LINK statement prior to being used on the ATMPVC statement.

When an ATM device is started, TCP/IP will attempt to activate all PVCs defined to all LINKs associated with the ATM device.

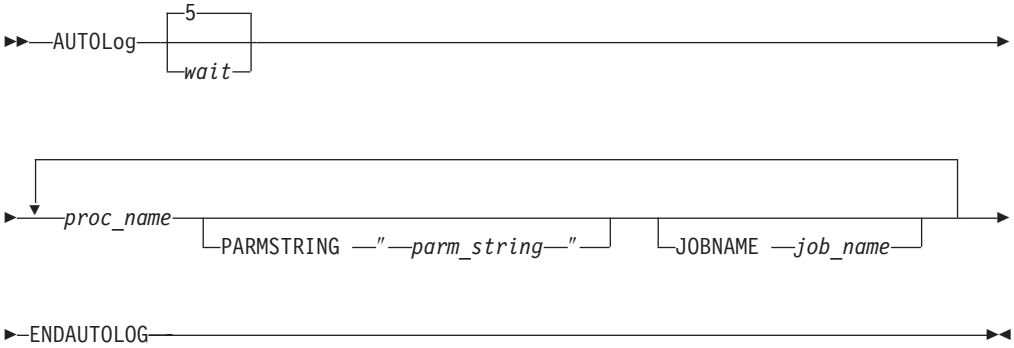
Related Topics

"DEVICE and LINK Statement—ATM Devices" on page 148

AUTOLOG Statement

Use the AUTOLOG statement to supply TCP/IP with the procedure names to start and the timeout value to use for a hung procedure during AUTOLOG. TCPIP starts the procedures in its current AUTOLOG list when it starts execution.

Syntax



Parameters

wait
The time TCP/IP should allow for a procedure to come down when at startup, it is still active and TCP/IP is attempting to AUTOLOG the procedure again. This could happen if the procedure did not come down when TCP/IP was last shutdown.

The default is 5 minutes. *wait* can be set to any finite time from 1 minute to 30 minutes. If a *wait* value outside the valid range is specified, the default of 5 minutes is used. When a *wait* value of 0 is specified, TCP/IP startup does not cancel and restart any procedures in the autolog list that are already started.

proc_name
A procedure that the TCP/IP address space should start. The procedure name must be a member of a cataloged procedure library.

PARMSTRING *"parm_string"*
A string to be added following the START procedure_name. Do not include the comma. The parm_string is 115 characters or less, not counting the double quotes around the string.

Note: The entire parm_string must be on one line.

JOBNAME *job_name*
The jobname used for the PORT reservation statement. This can be identical to the procedure_name, but for OS/390 UNIX jobs that spawn listener threads it is not. If the job_name is not explicitly set, it is assumed to be the same as the procedure_name.

Modifying

To modify the AUTOLOG statement, use a VARY TCPIP command with an OBEYFILE which contains a new AUTOLOG statement. The first AUTOLOG statement in an OBEYFILE replaces all previous AUTOLOG statements, and subsequent AUTOLOG statements in the same OBEYFILE append to the existing

statements. See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

Examples

This example shows how to include several servers in the AUTOLOG statement:

```
AUTOLOG
    FTPD JOBNAME FTPD1    ; FTP Server
    LPSERVE               ; LPD Server
    NAMESRV               ; Domain Name Server
    NCPROUT               ; NCPRoute Server
    PORTMAP               ; Portmap server
    OROUTED               ; RouteD Server
    RXSERVE               ; Remote Execution Server
    SMTP                  ; SMTP Server
    OSNMPD                ; SNMP Agent Server
    SNMPQE                ; SNMP Client Address space
    TCPIPX25              ; X25
    MVSNFS                ; Network File System Server
ENDAUTOLOG
```

This example shows how to autolog two procedures. The first is named MYPROC1. When it is started, it should use the following MVS console start command:

```
S MYPROC1,PARMS='-w 100',ID=XYZ
```

The second procedure has a listening OS/390 UNIX thread that is first spawned task. (You can use the MVS DISPLAY ACTIVE,LIST console command to determine the jobname.) If MYPROC21 abends or stops listening, the following MVS console start command is entered:

```
S MYPROC2,PARMS='-dzy 50',DSN='HLQ.'
```

```
AUTOLOG 20
    MYPROC1 PARMSTRING "PARMS='-w 100',ID=XYZ"
    MYPROC2 PARMSTRING "PARMS='-dzy 50',DSN='HLQ.'" JOBNAME MYPROC21
ENDAUTOLOG

PORT 2010 TCP MYPROC1
     2011 TCP MYPROC21
```

Usage Notes

- The ENDAUTOLOG statement specifies the end of the AUTOLOG parameters to pass to TCP/IP.
- TCPIP checks periodically to determine if each started task is still running. If a procedure is in the AUTOLOG list but there is no PORT statement for the procedure or there is a PORT statement but NOAUTOLOG is specified, TCP/IP will not attempt to automatically restart the procedure if it is currently not running.
- If the AUTOLOG list is empty, no additional services are started. If the AUTOLOG statement is omitted, TCP/IP does not attempt to start any of the higher-level services. Each service must be started manually.
- If a procedure 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 attempts to cancel that procedure and start it again. Kerberos is an example of a procedure that, if placed in the AUTOLOG list, would be subject to being periodically cancelled and restarted by TCP/IP because it does not have a permanent listening connection established.

  If the PORT statement specifies `NOAUTOLOG`, TCP/IP attempts to cancel the procedure but does not try to restart it.

  The INTCLIEN telnet task is never cancelled.

- The first AUTOLOG statement in a profile or obeyfile replaces all previous AUTOLOG statements. Subsequent statements in the same profile append to the existing statements.
- INTCLIEN cannot be specified as a procedure name on the AUTOLOG statement.

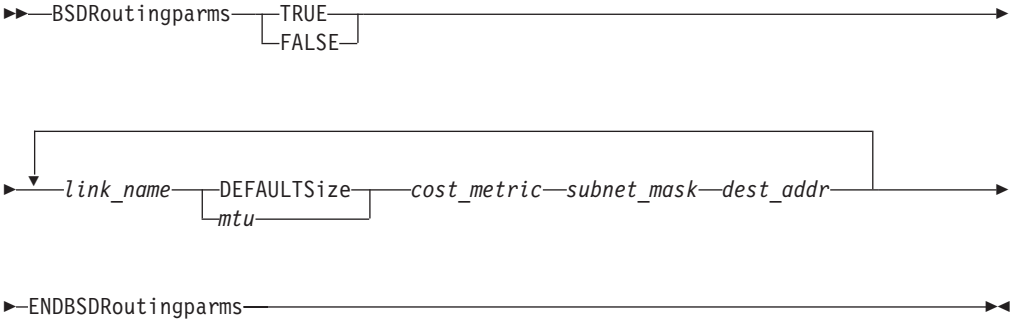
Related Topics

"PORT Statement" on page 229

BSDROUTINGPARMS Statement

Use the BSDROUTINGPARMS statement to define the characteristics of each link defined at the host over which OROUTED will send routing information to adjacent routers running the RIP protocol and which NCPROUTE will send transport PDUs to client NCPs.

Syntax



Parameters

TRUE
Specifies that the minimum packet size for the interface is always used regardless of whether the destination is one or more hops away.

FALSE
Specifies that the default maximum packet size of 576 is used (rather than the packet size of the interface) when sending to networks that are not locally attached.

link_name
The name of the link as defined in a LINK statement. Each link must be defined once in the BSDROUTINGPARMS statement.

mtu
Specify recommended values during later performance tuning:

- 1492 bytes for Ethernet 802.3
- 1500 bytes for Ethernet Version 2 IEEE
- 1500, 2000, or 4000 bytes (or greater if your environment permits) for token ring
- 2000 or 4000 bytes for FDDI. Use the larger value if permitted by your environment.
- 65527 bytes for CTC
- 4096 bytes for CLAW.

cost_metric
The metric associated with the cost of use for the link. When sending routing information over this link, OROUTED will add a metric value to the routing metrics for the routes that are to be broadcast over this link. The metric value that is added will be the value specified in the BSDROUTINPARMS section incremented by one. If a metric of zero is specified, a metric value of one will be added, which is the default cost for a directly-connected network. If a metric of one is specified, a metric value of two will be added. The higher metric

causes the route over this link to be less preferred. The range is from 0 to 14. A metric of 0 is usually coded so that the routes sent over the interface will be the most preferred.

subnet_mask
A bit mask (expressed in dotted-decimal form) having bits in the network or host portions that defines the subnet mask associated with the link. Depending upon the VARSUBNETTING setting in the ASSORTEDPARMS or IPCONFIG statement, the subnet mask can be in several forms as follows:

- NOVARSUBNETTING

  The network portion and the host portion making up the subnet must be bit-contiguous from left to right. If the link to the network is not subnetted, then the subnet_mask can be a network class mask. If the subnet_mask equals 0, the default will be the network class mask. Also, a single, fixed-length subnet mask must be defined for each network, that is, multiple subnets having the same network number must have identical subnet masks.
- VARSUBNETTING

  Variable-length subnet masks may be used in a single network, that is, multiple subnets having the same network number can have different subnet masks. The bits must be contiguous from left to right. A subnet mask that is less than the network class mask is considered to be a supernet mask. A supernet mask can be defined such that multiple networks can be represented by a single supernet. With a supernet mask, a supernet route can be defined to represent multiple network routes.

Note: The host mask of 255.255.255.255 cannot be used for the subnet mask.

dest_addr
Destination address applies to point-to-point links only. A non-zero destination address applies to non-broadcast and non-multicast capable point-to-point links. If zero is coded, directed broadcast or multicast address will be used; otherwise, insert the address of the host on the other end of the link. For VIPA links, this field should be zero. The following link types are defined as point-to-point: CDLC, CLAW, CTC, MPCPTP, SNALINK, SNALU62, and NPSI X.25.

Modifying

To modify the BSDROUTINGPARMS statement for a link, use a VARY TCPIP command with an OBEYFILE which defines a new BSDROUTINGPARMS statement for a link with the same link_name. The new BSDROUTINGPARMS statement is a complete replacement for the original BSDROUTINGPARMS statement. If you have changed the link's IP address, or the order of the HOME list entries, along with the BSDROUTINGPARMS changes, remember to include the new HOME list statement in the same OBEYFILE data set as the new BSDROUTINGPARMS statement. See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

Note: If you are using OROUTED, refer to Table 10 on page 142 for additional instructions.

Table 10. BSDROUTINGPARMS Modification Methods

Modification Method	Required Action
Adding new links	Issue VARY TCPIP,,OBEYFILE command with new DEVICE, LINK, HOME, and BSDROUTINGPARMS statements. No action is required for ORouteD.
Deleting or changing order of links in use by OROUTED.	Issue VARY TCPIP,,OBEYFILE command with new HOME statement. No action is required for ORouteD.
Changing HOME IP addresses or BSDROUTINGPARMS values for existing links in use by OROUTED	1. 1. Issue VARY TCPIP,,OBEYFILE command with new HOME and/or BSDROUTINGPARMS statments. 2. 2. Issue the MODIFY <OROUTED procname>,RECONFIG command.

Examples

This example shows the BSDROUTINGPARMS statement for several types of LAN media.

```
;  link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
   TR1       2000       0     255.255.255.0   0
   ETH1      1500       0     255.255.255.0   0
   FDDI1  DEFAULTSIZE   0     255.255.255.0   0
ENDBSDROUTINGPARMS
```

This example includes a link, LINK3, that is a point-to-point link between host MVS1 and host 128.84.54.6.

```
;
;  link     maxmtu    metric    subnet_mask    dest_addr
BSDROUTINGPARMS false
   LINK1  DEFAULTSIZE    0      255.255.255.0     0
   LINK2  DEFAULTSIZE    0      255.255.255.0     0
   LINK3  1500           0      255.255.255.0     128.84.54.6
ENDBSDROUTINGPARMS
```

This example shows the definitions for VIPA links.

```
BSDROUTINGPARMS false
   VLINK1   DEFAULTSIZE  0  255.255.255.252  0
   VLINK2   DEFAULTSIZE  0  255.255.255.252  0
ENDBSDROUTINGPARMS
```

This example shows how BSDRoutingparms relate to other statements in the profile.

```
DEVICE DEVC00   CTC C00 IOBUFFERSIZE 65535 AUTORESTART
LINK   LCTCC00  0   DEVC00 IFSPEED 10000
HOME   9.32.2.1 LCTCC00

DEVICE DEVD00   LCS D00 AUTORESTART
LINK   ETHERD00  ETHERNET 0 DEVD00
LINK   IBMTRD00  IBMTR    1 DEVD00
LINK   FIDDID00  FDDI     2 DEVD00
HOME   130.80.0.1 ETHERD00
       130.81.0.2 IBMTRD00
       130.82.0.3 FIDDID00

PRIMARYINTERFACE LCTCC00

BSDROUTINGPARMS TRUE
```

```
  LCTCC00  DEFAULTSIZE 0  255.252.0.0   9.32.2.5
  ETHERD00 DEFAULTSIZE 0  255.252.0.0   0
  IBMTRD00 DEFAULTSIZE 0  255.252.0.0   0
  ENDBSDROUTINGPARMS


IPCONFIG VARSUBNETTING

START  DEVD00
```

Usage Notes

- Use the BSDROUTINGPARMS statement whenever you are running OROUTED or NCPROUTE. In this case, define each link in the BSDROUTINGPARMS statement that is to be known by these services. For OMPROUTE, it is not necessary to define the BSDROUTINGPARMS statement.

  Note: If using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS statement is required to route Transport PDUs prior to OMPROUTE activation. Because the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both the BSDROUTINGPARMS statement and the OMPROUTE configuration file.
- The MTU value specified on BSDROUTINGPARMS statement will also be used for applications that use the setsockopt() IP_MULTICAST_IF option to specify the route for multicast datagrams.
- For rules on defining virtual IP addresses for VIPA links, see the “HOME Statement” on page 207.
- The maximum transmission unit (MTU) and metric of any other links with a destination address in the same subnet are updated to ensure that all entries in the same subnet have the same routing values. Except for these links and the LOOPBACK link, all links get default BSD values if not specified.
- The *subnet_mask* is related to the HOME IP address of the link. A ClassB HOME IP address (128.0.0.0 thru 191.255.255.255) cannot have a ClassA *subnet_mask* (255.0.0.0 thru 255.254.0.0) without IPCONFIG VARSUBNETTING set on. Similarly, a ClassC HOME IP address (192.0.0.0 thru 224.255.255.255) cannot have a ClassA or ClassB *subnet_mask* (255.0.0.0 thru 255.255.254.0) without IPCONFIG VARSUBNETTING set on.
- If no HOME address exists for a LINK or if a HOME address changes via a later VARY TCPIP, processing of the HOME statement will verify if the BSDROUTINGPARMS *subnet_mask* is within the valid ranges depending on the IPCONFIG VARSUBNETTING settings. OROUTED will ignore links without HOME addresses.
- If running Enterprise Extender with OROUTED, see “Configuring ORouteD with Enterprise Extender” on page 940.

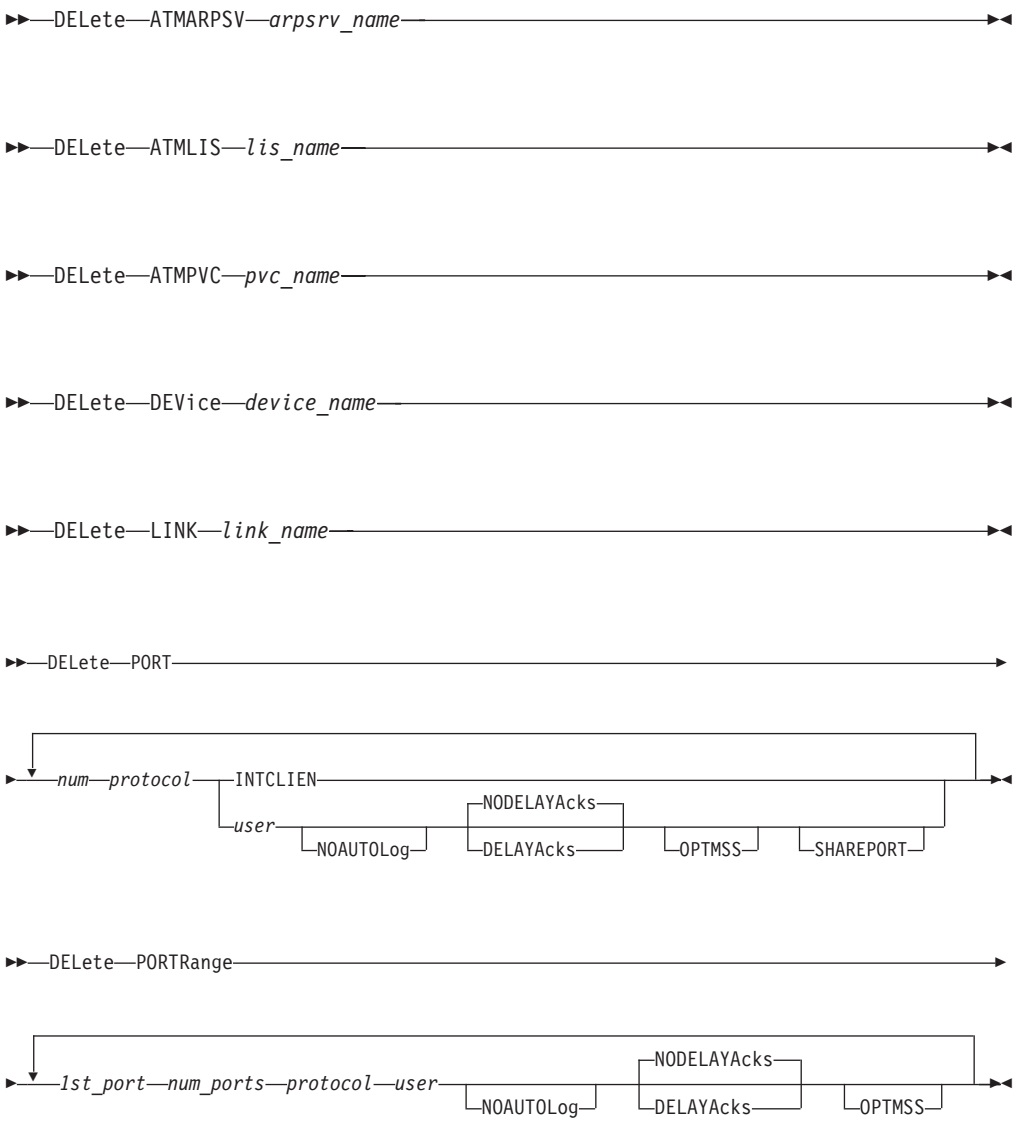
Related Topics

- “GATEWAY Statement” on page 193
- “VARY Command—TCPIP Address Space” on page 264
- “IPCONFIG Statement” on page 213
- “HOME Statement” on page 207
- “DEVICE and LINK Statement—Virtual Devices” on page 188

DELETE Statement

Use the DELETE statement to delete a previously defined ATMARPSV, ATMLIS, or ATMPVC device, link, port, or portrange.

Syntax



Parameters

arpsrv_name
The name of the ATMARP server to be deleted. This is the name that was used on an ATMARPSV statement to define the ATMARP server to TCP/IP.

lis_name
The name of the LIS to be deleted. This is the name that was used on an ATMLIS statement to define the LIS to TCP/IP.

pvc_name
The name of the PVC to be deleted. This is the name that was used on an ATMPVC statement to define the PVC to TCP/IP.

device_name
The name of the device to be deleted. This is the name that was used on a DEVICE statement to define the device to TCP/IP.

link_name
The name of the link to be deleted. This is the name that was used on a LINK statement to define the link to TCP/IP.

num
The port number of the port to be deleted. This is the port number that was used on a PORT statement to define the port to TCP/IP.

protocol
Specifies the protocol to be used, either TCP or UDP.

INTCLIEN
This keyword indicates the port is assigned to the internal telnet server rather than to a client. Therefore, you must use the same port number as specified on the TELNETPARMS statement. INTCLIEN is only allowed with the TCP protocol.

user
The job name associated with the port to be deleted.

NOAUTOLOG
Tells the TCP/IP address space ***not*** to restart the server if it was stopped previously.

DELAYACKS
Allows you to alter the default TCP/IP behavior for acknowledgements and delay their transmission so that they can be combined with data sent to the foreign host. This affects acknowledgements returned when a packet is received with the PUSH bit on in the TCP header. The default behavior is to return an acknowledgement immediately.

NODELAYACKS
Specifies that an acknowledgement is returned immediately. This is the default.

OPTMSS
Specifies that the optimal maximum segment size function is available. OPTMSS is only valid for TCP ports.

SHAREPORT
Required when reserving a port to be shared across multiple listeners.

1st_port
The first port number of the port range to be deleted. This is the same starting port number used on a PORTRANGE statement to define the port range to TCP/IP.

num_ports
The number of ports to be deleted starting from the *1st_port*. This is the same number of ports that were reserved when the port range was defined with the PORTRANGE statement.

Modifying

Modification is not applicable to this statement.

Examples

This example shows DELETE statements to delete an ATM PVC named PVC1, an ATM LIS named LIS1, and an ATMARPSV named ARPSV1:

```
DELETE ATMPVC PVC1
DELETE ATMLIS LIS1
DELETE ATMARPSV ARPSV1
```

This example shows DELETE statements that delete a link called sanjose and a device called ourctc:

```
DELETE LINK sanjose
DELETE DEVICE ourctc
```

This example shows a PORT statement to reserve port 5001 for MEGA, and then a DELETE PORT statement to delete the reservation of that port:

```
PORT 5001 TCP MEGA
DELETE PORT 5001 TCP MEGA
```

This example shows several PORTRANGE statements to reserve ports for MEGA, and then several DELETE PORTRANGE statements to delete the reservations for those ports:

```
PORTRANGE 5000 10 UDP MEGA
          5100 10 TCP MEGA NOAUTOLOG
          5200 10 UDP MEGA DELAYACKS
          5300 10 TCP MEGA
          5400 10 UDP MEGA
          5500 10 TCP MEGA NOAUTOLOG DELAYACKS
DELETE PORTRANGE
          5000 10 UDP MEGA
          5100 10 TCP MEGA NOAUTOLOG
          5200 10 UDP MEGA DELAYACKS
          5300 10 TCP MEGA
          5400 10 UDP MEGA
          5500 10 TCP MEGA NOAUTOLOG DELAYACKS
```

Usage Notes

- To delete a link, you must first delete any associated HOME entry by specifying a HOME statement that does not include the link, and you must also stop the device.
- To delete an ATM link, you must first delete any associated ATMPVCs.
- To delete an ATMLIS, you must first delete all associated LINKs and ATMARPSVs.
- To delete an ATMARPSV, you must first stop all devices which have a LINK associated with the ATMLIS for the ATMARPSV.
- You can delete an ATMPVC for a started device. However, if the PVC is in use as an ATMARP server, you must first stop the device(s) using the PVC as an ATMARP server in order to delete the ATMPVC.
- To delete a device, you must first delete all associated links.
- To delete a device that is started, you must first stop the device.
- Individual ports cannot be removed from a PORTRANGE; you must delete the entire PORTRANGE.
- To delete a specific PORT or PORTRANGE, the port number/range, protocol and user value of the corresponding PORT or PORTRANGE statement must also be specified on the DELETE PORT or DELETE PORTRANGE statement. They are not checked, but are supported so that the PORT and PORTRANGE statements can be reused by placing the statement DELETE before them.

- Because VIPA devices can not be started or stopped and are always active, a VIPA link can not be deleted.

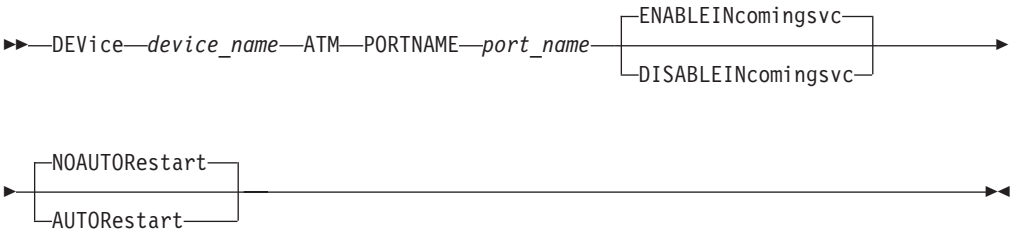
DEVICE and LINK Statement—ATM Devices

Use the DEVICE statement to specify the name of the asynchronous transfer mode (ATM) device that you use. Use the LINK statement to define a network interface link associated with the ATM device.

The presence of DEVICE and LINK ATM statements in your *PROFILE.TCPIP* enables ATM native mode and SNMP network management support for the ATM device. Even if an ATM device is not being used by this TCP/IP, or is being used by TCP/IP in an ATM LAN Emulation mode instead of Native mode, specifying DEVICE and LINK statements will enable you to retrieve SNMP network management data for the ATM device. Enablement of SNMP network management data for the ATM devices also requires specification of the ATMENABLED parameter on the SACONFIG Profile statement.

You can specify multiple LINKs for an ATM device. This is so an ATM device can be in more than one LIS.

Syntax



Parameters

device_name
The name of the device. The device name must be the OSA name known to MPC and OSA/SF. The maximum length is 8 characters. This name, the OSA name, must match the name specified on the transport resource list element (TRLE). For more information on the TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference.* The same name is specified in the LINK statement(s).

ATM
Specifies the device is for ATM use.

PORTNAME *port_name*
The OSA port name. The maximum length is 8 characters. This name must match the portname specified on the transport resource list element (TRLE). For more information on the TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

DISABLEINcomingsvc
Device cannot be used for incoming SVCs.

ENABLEINcomingsvc
Allow incoming SVC calls for this device; the device can be used for both outgoing and incoming SVCs.

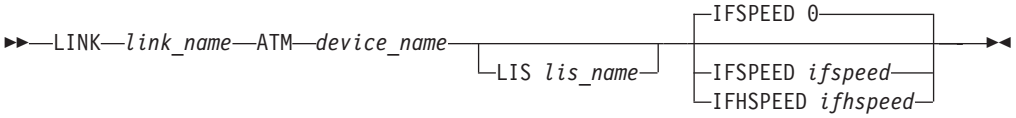
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

ATM
Specifies that the link is an ATM link.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

LIS *lis_name*
The logical IP subnet for this LINK. This parameter is only required if the link is to be used for SVC connections. The maximum length is 16 characters, padded with blanks. The *lis_name* must be defined on an ATMLIS statement prior to being used on the LINK statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added

4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

Examples

The following example specifies that OSA1 is an ATM device:

```
DEVICE OSA1  ATM  PORTNAME  PORT1
LINK   LINK1 ATM  OSA1
```

Usage Notes

- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- When an ATM device is started, TCP/IP updates the IFSPEED and IFHSPEED values with the actual link speed of the interface.
- To use dynamic routing with this device, see “Configuring NBMA Subnetworks” on page 1029.

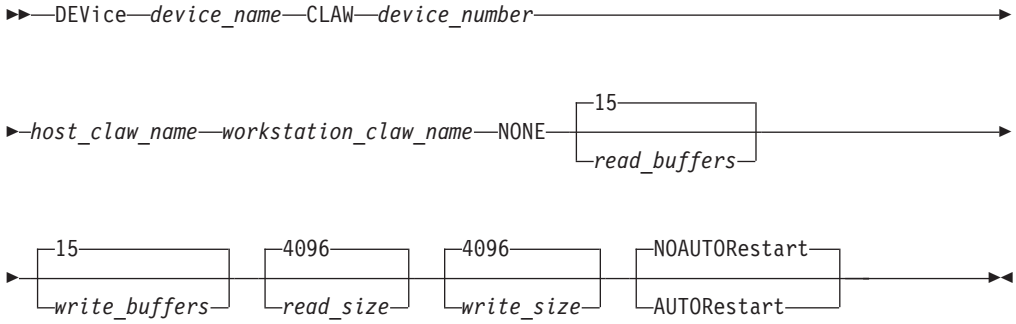
Related Topics

- “DEVICE and LINK Statements” on page 101
- “Chapter 22. Configuring Simple Network Management Protocol (SNMP)” on page 823
- “BSDROUTINGPARMS Statement” on page 140
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “VARY Command—TCPIP Address Space” on page 264
- “TRANSLATE Statement” on page 249
- “ATMARPSV Statement” on page 131
- “ATMLIS Statement” on page 133
- “ATMPVC Statement” on page 136

DEVICE and LINK Statement—CLAW Devices

Use the DEVICE statement to specify the name and hexadecimal device number of a Common Link Access to Workstation (CLAW) device that you use. Devices that use the CLAW protocol include RISC System/6000 and SP2®. Only one DEVICE statement should be used for each device. Use the LINK statement to define a network interface link associated with CLAW devices. Only one LINK statement should be used for each device.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CLAW
Specifies the device is a CLAW device.

device number
The hexadecimal device number of the RISC System/6000. TCP/IP also uses device number + 1.

host_claw_name
A value that defines the name of the host system in the system validation exchange between the TCP/IP code and the workstation code. This name must be the same name that is defined as the CLAW mode HOST name in SMIT on the RISC/6000.

The maximum length is 8 characters.

workstation_claw_name
A value for the name of the workstation for the system validation exchange. This value must be the name of the CLAW mode adapter defined in SMIT on the RISC/6000. The maximum length is 8 characters.

NONE
A place holder reserved for future use.

read_buffers
This is the decimal number (one or more) of buffers to allocate to the read channel program. This should be large enough to give TCP/IP sufficient time to process the received data and append the buffer to the running channel

program before it terminates. Each of these buffers uses real storage, so the number should be small enough not to impact overall system performance. The default is 15.

write_buffers
This is the decimal number (one or more) of buffers to allocate to the write channel program. This should be large enough that a busy TCP/IP can reuse buffers without the channel program terminating. Each of these buffers uses real storage, so the number should be small enough not to impact overall system performance. The default is 15.

read_size
This is the size of the read buffers. Values are:
1024
2048
3072
4096

The value must be greater than or equal to the transmit buffer size specified in the RISC System/6000. The default is 4096. The value must be 4096 for ESCON® or RISC System/6000.

write_size
This is the size of the write buffers. Values are:
1024
2048
3072
4096

The value must be less than or equal to the receive buffer size specified in the RISC System/6000. The default is 4096. The value must be 4096 for ESCON or RISC System/6000.

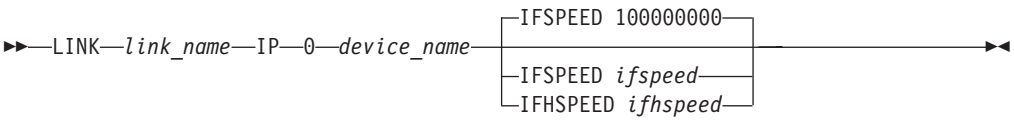
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCPIP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

IP
A constant.

0 A constant.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to includeexisting links that you want to have active in your configuration.

Examples

This example shows how you might code DEVICE, LINK, and related statements for a RISC System/6000 connection.

```
DEVICE RS6K CLAW 6B2 HOST PSCA NONE
LINK IPLINK1 IP 0 RS6K
HOME
   192.10.10.1 IPLINK1

GATEWAY
```

```
;
; Network  First hop   Driver   Packet size  Subnet mask  Subnet value

  192.10.10.2  =         IPLINK1 DEFAULTSIZE  HOST
  DEFAULTNET 192.10.10.2 IPLINK1 DEFAULTSIZE  0

; The following BSDROUTINGPARMS statement would be used if running OROUTED
;
;   link     maxmtu   metric   subnet mask     dest addr
 BSDROUTINGPARMS false
    IPLINK1   2000       0     255.255.255.0   192.10.10.2
 ENDBSDROUTINGPARMS
 ;

START RS6K
```

Usage Notes

- If the OS/390 server running the CLAW device driver is a second-level host on a first-level VM system, the OS/390 server should be defined as V=R (virtual=real). Unpredictable results might occur if CLAW is run without V=R, as dynamic changes to the channel program (performed by the CLAW driver) will not be observed by the channel.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP variables are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.

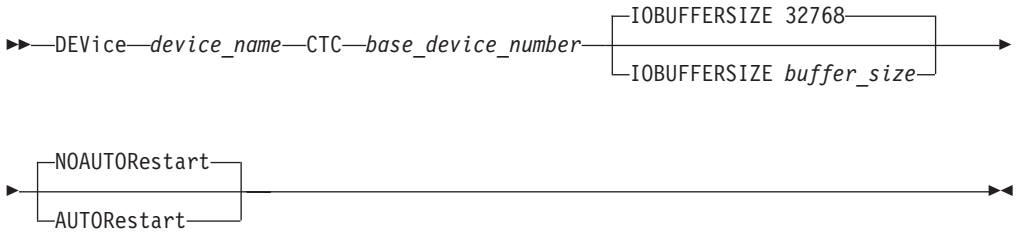
Related Topics

- “BSDROUTINGPARMS Statement” on page 140
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “VARY Command—TCPIP Address Space” on page 264

DEVICE and LINK Statement—CTC Devices

Use the DEVICE statement to specify the name and hexadecimal device number of the channel-to-channel (CTC) devices that you use. Use the LINK statement to define a network interface link associated with the CTC devices. You must use a separate DEVICE statement for each device you use. The same is true for the LINK statement.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CTC
Specifies the device is a channel-to-channel (CTC) device.

base_device_number
The hexadecimal base device number associated with the CTC adapter. Two numbers are used by TCP/IP: the *base_device_number* and *base_device_number+1*.

IOBUFFERSIZE *buffer_size*
Specifies the I/O buffer size. The buffer size must be 32K, 32768, or 65535.

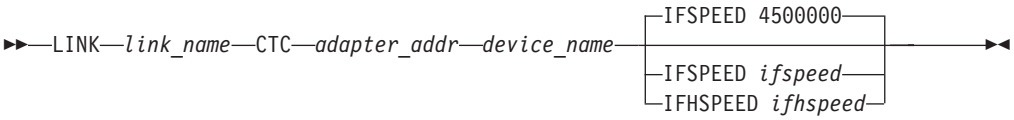
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

CTC
Specifies that the link is a channel-to-channel link.

adapter_addr
An integer used to specify which device number is the read device number and which device number is the write device number. Use 0 to indicate that the base device number is the read device and 1 to indicate that the base device number is the write device.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Usage Notes

- The configured I/O buffer sizes at each end of the CTC connection must match. A buffer size mismatch can cause packet loss or I/O errors, resulting in deactivation of the CTC connection. CTC I/O buffer size may be explicitly specified with the IOBUFFERSIZE parameter.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.

Related Topics

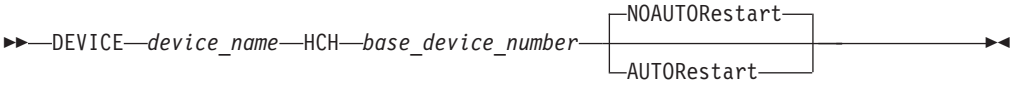
- “BSDROUTINGPARMS Statement” on page 140
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “STOP Statement” on page 246
- “VARY Command—TCPIP Address Space” on page 264

DEVICE and LINK Statement—HYPERchannel A220 Devices

Use the DEVICE statement to specify the name and hexadecimal device number of the HYPERchannel A220 device.

Use the LINK statement to define the link to the HYPERchannel A220 adapter.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

HCH
Specifies the device is a HYPERchannel A220.

base_device_number
The hexadecimal base device number associated with the A220 adapter. Two addresses are used by TCP/IP: the *base_device_number* and *base_device_number+1*.

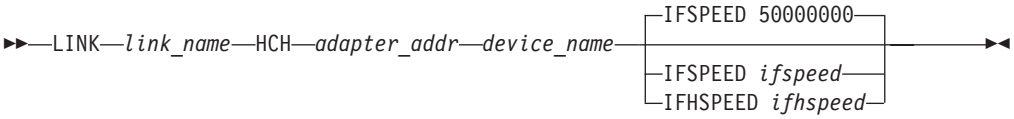
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

HCH
Specifies that the link is a HYPERchannel A220.

adapter_addr
Must be an integer, but the 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 same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Usage Notes

- The ATTENTION+BUSY and unit check conditions are normally handled in the background and can affect performance without any visible evidence. The recommendations on HYPERchannel A222 and A223 Mode Switch Settings are:
  - The *Disable Attentions* setting on the HYPERchannel box eliminates the ATTENTION+BUSY status in response to read commands, which reduces overhead.
  - The *Enable Command Retry* setting reduces the number of unit checks needed because of trunk contention. This setting improves performance,

because the TCP/IP device driver waits 10 milliseconds before retrying a command that produced a unit check. This setting also eliminates the need to perform sense operations and retry commands.

  - To use dynamic routing with this device, see “Configuring NBMA Subnetworks” on page 1029.

- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- The TRANSLATE statement is required for HYPERchannel A220 devices.

Related Topics

- “BSDROUTINGPARMS Statement” on page 140
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “STOP Statement” on page 246
- “VARY Command—TCPIP Address Space” on page 264
- “TRANSLATE Statement” on page 249

DEVICE and LINK Statement—LAN Channel Station and OSA Devices

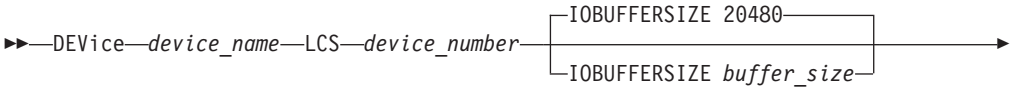
Use the DEVICE statement to specify the name and hexadecimal device number of an IBM 8232 LAN channel station (LCS) device, an IBM 3172 Interconnect Controller, an IBM 2216 Multiaccess Connector Model 400®, an IBM FDDI, Ethernet, Token Ring OSA, or an IBM ATM OSA-2 in LAN emulation mode.

Use the LINK statement to define a network interface link associated with an LCS device. The LINK statements used are the Ethernet Network LCS LINK statement, the Token-Ring Network or PC Network LCS LINK statement, and the FDDI LCS LINK statement.

You must use a separate LINK statement for each link associated with an LCS device.

Note: Each network interface on the OSA adapter is considered a separate DEVICE. For example, if you are using both ports on the OSA-2 card, you need to code a DEVICE and LINK pair for each port.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified on the LINK statements.

LCS
Specifies the device is a LAN Channel Station.

device_number
The hexadecimal device number of the LCS. *device_number*+1 is also used by the TCP/IP address space.

IOBUFFERSIZE *buffer_size*
Specifies the I/O buffer size. The buffer size must be 20K, 20480, 32K, or 32768.

Notes:

1. The configured I/O buffer sizes for the host and for the device must match. A buffer size mismatch can cause packet loss or I/O errors, which results in the deactivation of the LCS connection.
2. If the LCS device supports an option to configure a 32K buffer size, then configuring both the device and the TCP/IP profile to 32K will provide the best performance. If the device does not support this option, then specify (or default) to 20K in the TCP/IP profile.

NETMAN
Specifies that this device is a 3172 that supports the IBM Enterprise-specific MIB variables for 3172.

NONETMAN
Specifies that this device will not be used for NETMAN data retrieval.

AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

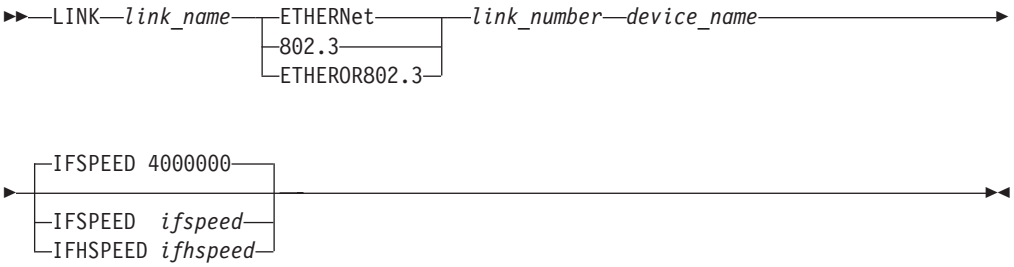
TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

LINK Statement for Ethernet Network LCS:

This LINK statement is used to define an Ethernet link on an IBM 3172 Interconnect Controller and IBM 8232 LAN Channel Station (LCS) device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

ETHERNET
Standard Ethernet protocol only.

802.3
IEEE 802.3 protocol only.

ETHERor802.3
Both standard Ethernet and IEEE 802.3 protocols.

Note: When ETHERor802.3 is specified, ARP packets for both protocols are generated. All devices on the network must be able to process or discard these packets.

link_number
The relative adapter number (0 for the first Ethernet protocol network in the LCS, 1 for the second Ethernet protocol network, and so on).

device_name
The *device_name* must be the same name as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

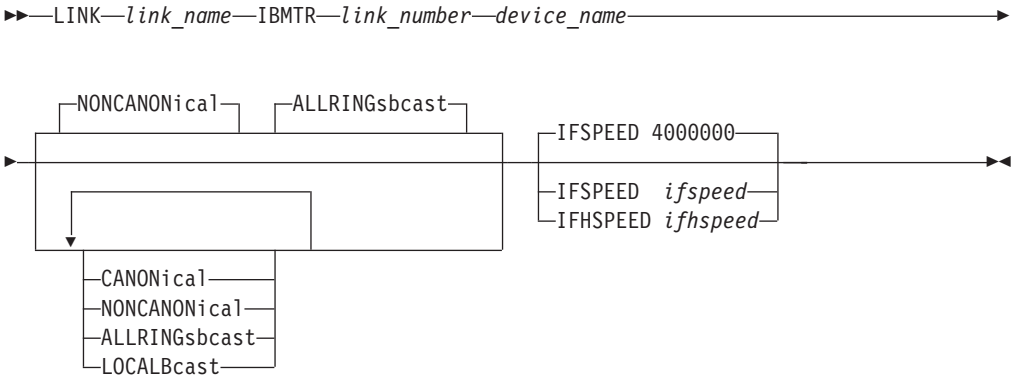
LINK Statement for Token-Ring Network or PC Network LCS:

The token-ring LCS LINK statement is used to define the token-ring link to the LCS (IBM 8232 or IBM 3172) previously defined by the LCS DEVICE statement. By default, the token-ring LCS LINK statement is also used to define the PC Network link.

Medium Access Control (MAC) addresses in the Address Resolution Protocol (ARP) packets on this token-ring network are in the more common, non-canonical format.

Note: All TCPIP hosts and gateways on a given token-ring network must be configured to use the same form for MAC addresses in ARP packets, either canonical or non-canonical. For more information about the terms, canonical and non-canonical, see IEEE standards 802.3 and 802.5.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

IBMTR
Specifies that the link is to an IBM Token Ring.

link_number
The relative adapter number (0 for the first token ring adapter in the LCS, 1 for the second token ring, and so on).

device_name
The *device_name* must be the same as specified in the DEVICE statement.

CANONICAL
MAC addresses in Address Resolution Protocol (ARP) packets on this token-ring network are in the canonical IEEE 802.5 form.

NONCANONICAL
MAC addresses in ARP packets on this token-ring network are in the more common non-canonical format. This is the default.

ALLRINGSBCAST
All IP and ARP broadcasts are sent as all-rings broadcasts, which are propagated through token-ring bridges (Source Route Bridging). This is the default.

LOCALBCAST
All IP and ARP broadcasts are sent only on the local ring and are not propagated through token-ring bridges (Transparent Bridging).

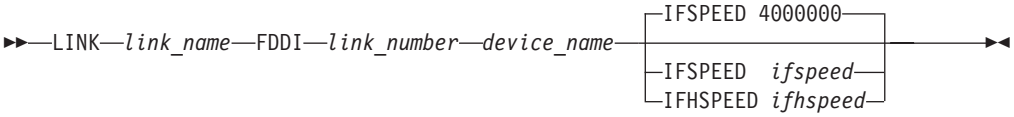
IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

LINK Statement for FDDI LCS:

This LINK statement is used to define the Fiber Distributed Data Interface (FDDI) link to the LCS (IBM 3172 Models 002 and 003) defined by the LCS DEVICE statement.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

FDDI
Specifies that the link is to an FDDI network.

link_number
The relative adapter number (0 for the first FDDI adapter in the LCS, 1 for the second FDDI adapter, and so on).

device_name
The *device_name* must be the same as specified in the DEVICE statement.

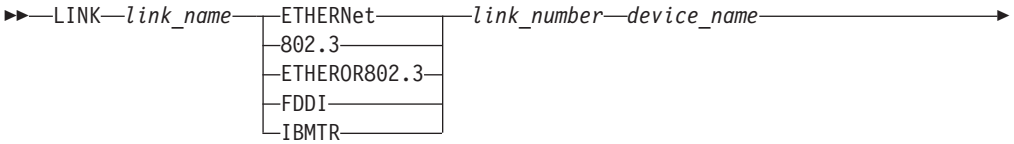
IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

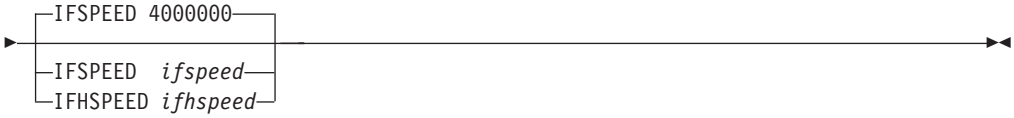
IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

LINK Statement for OSA:

This LINK statement is used to define a network interface link associated with an OSA device.

Syntax





Parameters

link_name
The name of the link. The maximum length is 16 characters.

ETHERNET
Standard Ethernet protocol only.

802.3
IEEE 802.3 protocol only.

ETHERor802.3
Both standard Ethernet and IEEE 802.3 protocols.

Note: When ETHERor802.3 is specified, ARP packets for both protocols are generated. All devices on the network must be able to process or discard these packets.

FDDI
Specifies that the link is to an FDDI network.

IBMTR
Specifies that the link is to an IBM token ring.

link_number
The port number on the OSA adapter.

device_name
The *device_name* must be the same name as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Examples

- In this example, LCS1 is a 3172 model 1 with a Token Ring and Ethernet adapter.

```
DEVICE LCS1   LCS           BA0
LINK TR1   IBMTR    0 LCS1
LINK ETH1  ETHERNET 1 LCS1
```

- In this example, LCS2 is a 3172 model 2 with a FDDI adapter.

```
DEVICE LCS2   LCS           BE0
LINK FDDI1 FDDI     0 LCS2
```

- This example shows how you might code DEVICE, LINK, and related statements for an LCS connection.

```
DEVICE LCS1 LCS BA0
LINK TR1  IBMTR    0 LCS1
LINK TR2  IBMTR    1 LCS1 LOCALBCAST
LINK ETH1 ETHERNET 0 LCS1
HOME
   192.10.10.10  TR1
   9.67.43.10    TR2
   128.50.17.1   ETH1

   GATEWAY
;
; Network  First hop   Driver   Packet size  Subnet mask  Subnet value

  192.10.10    =         TR1       2000       0
  9            =         TR2       2000       0.255.255.0  0.67.43.0
  128.50       =         ETH1      1500       0.0.240.0    0.0.16.0
  DEFAULTNET 9.67.43.1   TR2    DEFAULTSIZE   0

; The following BSDROUTINGPARMS statement would be used if running OROUTED.
; If not running OROUTED, use prior gateway stats.
;
; ;  link     maxmtu   metric   subnet mask     dest addr
; BSDROUTINGPARMS false
;    TR1       2000       0     255.255.255.0   0
;    TR2       2000       0     255.255.255.0   0
;    ETH1      1500       0     255.255.240.0   0
; ENDBSDROUTINGPARMS
;

START LCS1
```

- In this example of an OSA-2 card, LCS1 is Token Ring Port 0 and LCS2 is an ETHERNET Port 1.

```
DEVICE LCS1   LCS           BA0
LINK TR1   IBMTR    0 LCS1
DEVICE LCS2   LCS           BAO
LINK ETH1  ETHERNET 1 LCS1
```

Usage Notes

- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- MPCOSA devices cannot be configured to accept IP packets destined to an IP address other than the IP address of the OSA-2 adapter. For example, IP packets destined to a Virtual IP Address (VIPA) owned by this TCP/IP will not be delivered by the OSA-2 adapter.
- MPCOSA devices do not support Multicast or Broadcast.
- No dynamic routing protocol (RIP, OSPF) can be communicated over MPCOSA devices. If routes are desired for resources that can be reached via this device, they must be statically defined.
- MPCOSA devices have an ARP offload function which offloads all ARP processing to the OSA-2 adapter. TCP/IP cannot display any ARP cache information or ARP counter statistics for these devices because OSA-2 does not provide this data to TCP/IP.

Related Topics

- “BSDROUTINGPARMS Statement” on page 140
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207

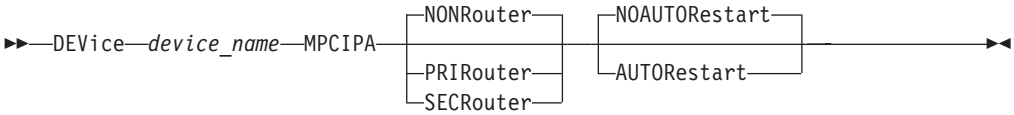
- “START Statement” on page 244
- “VARY Command—TCPIP Address Space” on page 264

DEVICE and LINK Statement—MPCIPA Devices

When defining an MPCIPA device, use the DEVICE statement to specify the PORT name contained in the TRLE definition for the QDIO interface. The TRLE must be defined as MPCLEVEL=QDIO. For details on defining a TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

Use the LINK statement to define a network interface link associated with the QDIO interface. Only one LINK statement can be specified for each MPCIPA device.

Syntax



Parameters

device_name
The name of the device. The device name must be the PORT name of the LAN adapter defined in a TRLE for a QDIO connection. The maximum length is 8 characters.

MPCIPA
Specifies the device belongs to the MPC family of interfaces and uses the IP Assist based interface. If a datagram is received at this device for an unknown IP address, the device will route the datagram to the TCP/IP instance defined as PRIROUTER. If there is no active TCP/IP instance defined as PRIROUTER, the device will route the datagram to the TCP/IP instance defined as SECROUTER. If there is no active TCP/IP instance defined as PRIROUTER or SECROUTER, the device will discard the datagram.

NONROUTER
If a datagram is received at this device for an unknown IP address, the datagram will not be routed to this TCP/IP instance.

PRIROUTER
If a datagram is received at this device for an unknown IP address, the datagram will be routed to this TCP/IP instance.

SECROUTER
If a datagram is received at this device for an unknown IP address and there is no active TCP/IP instance defined as PRIROUTER, then the datagram will be routed to this TCP/IP instance.

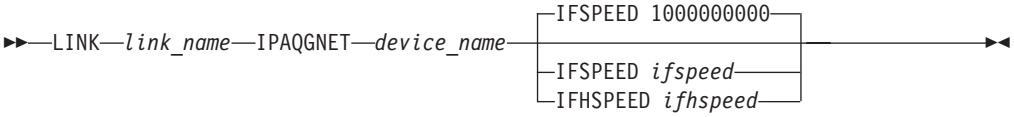
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 2 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

IPAQGNET
Indicates that the link uses the IP Assist based interface, belongs to the QDIO family of interfaces, and uses the Gigabit Ethernet protocol.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Usage Notes

- When you start an MPCIPA device, TCP/IP registers the entire set of local (home) IP addresses for this TCP/IP instance to OSA-Express. This allows the device to route datagrams destined for those IP addresses to this TCP/IP instance. (If the device receives a datagram destined for an unregistered IP address, then OSA-Express sends the datagram to the TCP/IP instance that is defined as the primary router or secondary router for this device.)

  If you subsequently add, delete, or change any home IP addresses on this TCP/IP instance, TCP/IP dynamically registers the changes to OSA-Express.
- For detailed instructions on setting up an OSA-Express feature, see the *S/390 Open Systems Adapter-Express Customer's Guide and Reference*.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- Across one central processor complex (CPC)[1], PRIROUTER and SECROUTER can only be specified in the profile of one TCP/IP instance for the same MPCIPA device. If PRIROUTER or SECROUTER is specified for an MPCIPA device but was already specified for the same device in the profile of another active TCP/IP instance, a warning message will be issued during START DEVICE processing for the device.
- The following rules apply to the setting and operation of primary and secondary routers:
  - For each device, the PRIROUTER and SECROUTER parameters can only be in effect for one TCP/IP instance within a CPC. Only one TCP/IP instance can be the primary router for the device and only one can be the secondary router. If a second TCP/IP instance attempts to activate a device with the PRIROUTER or SECROUTER parameters, the specified parameter will be ignored for the second TCP/IP instance.

    There is no requirement that the same TCP/IP instance be specified PRIROUTER or SECROUTER for all OSA-Express adapters attached to the CPC.
- MPCIPA devices have an ARP offload function which offloads all ARP processing to the OSA-Express adapter. TCP/IP cannot display any ARP cache information or ARP counter statistics for these devices because OSA-Express does not provide this data to TCP/IP.

Related Topics

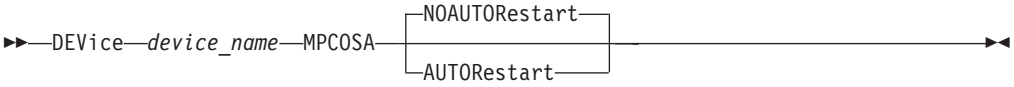
- "DEVICE and LINK Statements" on page 101
- "VARY Command—TCPIP Address Space" on page 264
- "GATEWAY Statement" on page 193
- "HOME Statement" on page 207
- "START Statement" on page 244
- "STOP Statement" on page 246

1. A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

DEVICE and LINK Statement—MPCOSA Devices

When defining a multipath channel MPCOSA connection, use the DEVICE statement to specify the TRLE name of an HPDT connection. Use the LINK statement to specify Fast Ethernet OSA-2 or FDDI OSA-2.

Syntax



Parameters

device_name
For MPCOSA connections, the device_name must be the name of the TRLE definition that corresponds to the OSA-2 configuration. You need to use OSA/SF to configure the OSA-2 to run in HPDT MPC mode. The TRLE is defined in a VTAM TRL major node and must be active to start the device. For details on defining a TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

The maximum length is 8 characters.

MPCOSA
Specifies that the device is a multipath channel MCPOSA device.

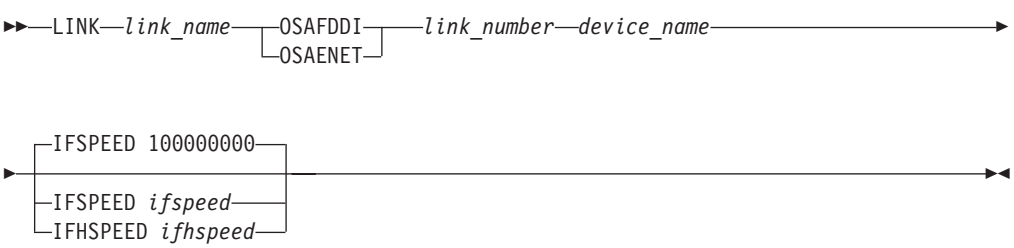
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 2 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters. The link name is associated with a home address on the HOME statement.

OSAFDDI
Specifies that the link is for MPCOSA FDDI OSA-2.

OSAENET
Specifies that the link is for MPCOSA Fast Ethernet OSA-2.

link_number
Specifies the OSA *link_number* of this interface and identifies the external attachment to a LAN supported by the OSA using the IP protocol.

Note: Zero (0) is the only link number supported.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interfaces current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interfaces current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY,

HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Usage Notes

- Virtual IP address (VIPA) cannot be used with this statement.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- MPCOSA devices have an ARP offload function which offloads all ARP processing to the OSA-2 adapter. TCP/IP cannot display any ARP cache information or ARP counter statistics for these devices because OSA-2 does not provide this data to TCP/IP.
- MPCOSA devices cannot be configured to accept IP packets destined to an IP address other than the IP address of the OSA-2 adapter. For example, IP packets destined to a Virtual IP Address (VIPA) owned by this TCP/IP will not be delivered by the OSA-2 adapter.
- MPCOSA devices do not support multicast or broadcast.
- To use dynamic routing with this device, see “Configuring NBMA Subnetworks” on page 1029.

Related Topics

- “DEVICE and LINK Statements” on page 101
- “VARY Command—TCPIP Address Space” on page 264
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “STOP Statement” on page 246

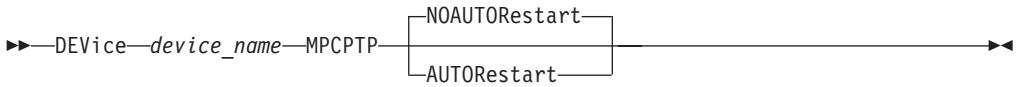
DEVICE and LINK Statement—MPCPTP Devices

When defining a High Performance Data Transfer (HPDT) connection, use the DEVICE statement to specify the name of the TRLE definition for the multipath channel (MPC) group. Also, the TRLE must be defined as MPCLEVEL=HPDT.

When defining an Enterprise Extender connection to the VTAM instance running on this host, use the DEVICE statement to define an IUTSAMEH interface. IUTSAMEH can also be used to define a connection between two TCP/IP stacks on the same system, and the MPCPTP device and link statements can be used to define XCF connections between two TCP/IP stacks in the same sysplex.

Use the LINK statement to define a network interface link associated with MPC group when defining an HPDT connection, or a network interface link associated with the IUTSAMEH interface when defining an Enterprise Extender connection.

Syntax



Parameters

device_name

For HPDT MPC connections to an IBM 2216 Multiaccess Connector Model 400, an IBM RS/6000®, or another OS/390 host, the device_name must be the TRLE name of an HPDT connection. The TRLE is defined in a VTAM TRL major node and must be active to start the device. For details on defining a TRLE, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*

The maximum length is 8 characters..

The reserved TRLE name IUTSAMEH can be used to bring up an MPCPTP connection between two TCP/IP stacks on the same system without the need for a physical device connection between the two stacks. The reserved TRLE name IUTSAMEH can also be used to define an Enterprise Extender connection to the VTAM instance running on this host. If you are defining an Enterprise Extender connection the device name must be IUTSAMEH.

Note: VTAM automatically activates the IUTSAMEH TRLE.

For XCF connections, the device_name must be the cpname of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active in both nodes to start the device. If connectivity is to a pre-R8 node, the XCF TRLE must also be active.

This value is also specified for device_name in the MPCPTP LINK statement.

MPCPTP

Specifies the device is a multipath channel point-to-point device.

AUTORESTART

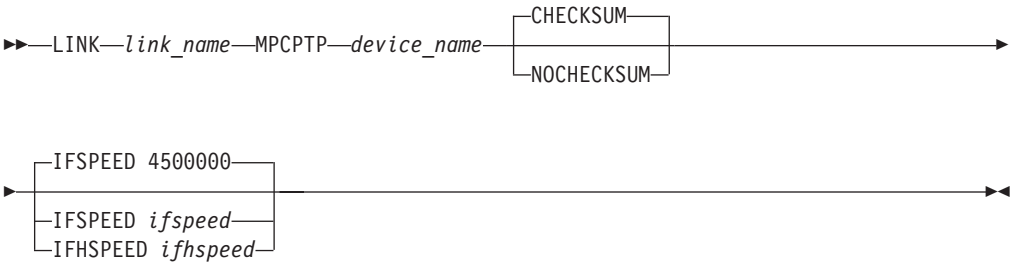
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 2 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters. The link name is associated with a home address on the HOME statement.

MPCPTP
Specifies that the link is for MPCPTP.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

CHECKSUM
Inbound checksum calculation will be performed for all packets received on this interface. This is the default.

NOCHECKSUM
Inbound checksum calculation will not be performed for any packets received on this interface.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:

- A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
- DELETE linkname and DELETE devicename statements

3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Usage Notes

- The CHECKSUM or NOCHECKSUM setting affects only the inbound TCP/IP data path. This setting has no effect upon the outbound path (checksum calculation is always performed outbound).

  While a performance gain can be achieved by specifying NOCHECKSUM, it is recommended that NOCHECKSUM be specified only for single-hop MPCPTP links (that is, where application traffic terminates in the adjacent node), such as S/390 to RS/6000 point-to-point connections. In such a configuration, the S/390 channel provides a reliable data path, thereby minimizing the need for TCP/IP checksum in detecting transmission errors.

  Note also that the TCP/IP checksum is useful in detecting software errors at the sending side, so it is further recommended that NOCHECKSUM be specified only when the sending-side software is considered reliable.
- An MPCPTP DEVICE and LINK statement is required if you plan to use the Enterprise Extender function and the TCP/IP stack you are configuring will be used for access to the IP network by VTAM on this host. For more information on configuring Enterprise Extender, see *OS/390 SecureWay Communications Server: SNA Network Implementation Guide*.
- For installations that plan on dedicating the MPC group for exclusive use by a single TCP/IP stack, improved performance will be achieved by explicitly defining the MPC group as MPCUSAGE=EXC. For additional information on the MPCUSAGE keyword, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.

Related Topics

- “DEVICE and LINK Statements” on page 101
- “VARY Command—TCPIP Address Space” on page 264
- “GATEWAY Statement” on page 193
- “HOME Statement” on page 207
- “START Statement” on page 244
- “STOP Statement” on page 246

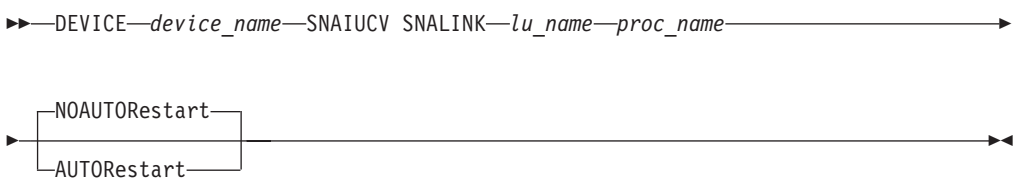
DEVICE and LINK Statement—SNA LU0 Links

Use the DEVICE statement to specify the name of the address space running the SNALINK program and the LU name of the 3745 Communications Controller to which an Ethernet or token ring is attached. These statements are required for NCPROUTE.

Use the LINK statement to define the link on the SNA LU type 0 DEVICE statement.

Note: Use this method to configure TCP/IP to access the 3745 adapter through SNALINK.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

SNAIUCV SNALINK
Specifies that the connection operates as an SNA LU type 0.

lu_name
The logical unit (LU) name of the remote end.

proc_name
The name of the SNALINK started procedure that runs on the host end.

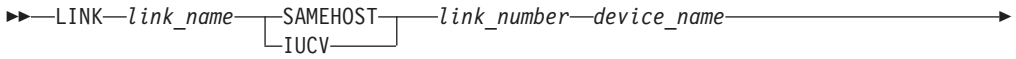
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCPIP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



There must be only one LINK statement for each SNA LU type 0 device statement.

Parameters

link_name
The name of the link. The maximum length is 16 characters.

SAMEHOST
Specifies that the DEVICE for SNA LU type 0 support uses a SAMEHOST connection.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

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 same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command but cannot modify any existing ones.

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 OBEYFILE data set.

Examples

In this example, SNALU0 is a SNA Link.

```
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINK
LINK SNA1 SAMEHOST     1 SNALU0
```

Usage Notes

- You can specify multiple LU0 DEVICE statements for the same SNALINK started procedure. A single LU0 address space can support multiple SAMEHOST links. A SAMEHOST link is created for each pair of LU0 DEVICE and LINK statements.

  However, you must specify a different *lu_name* for each DEVICE statement. This value is passed to the LU0 application to establish a session with a remote LU of that name.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.

Related Topics

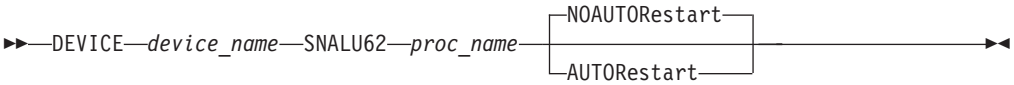
"VARY Command—TCPIP Address Space" on page 264

DEVICE and LINK Statement—SNA LU 6.2 Links

Use the DEVICE statement to specify the name of the started procedure running the SNALINK LU6.2 interface program.

Use the LINK statement to define the link to the SNALINK LU6.2 Interface program. There must be only one LINK statement for each SNA LU type 6.2 DEVICE statement.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

SNALU62
Specifies that the connection operates by a SNA LU type 6.2 session.

proc_name
The name of the SNALINK started procedure (on this node) that controls the device.

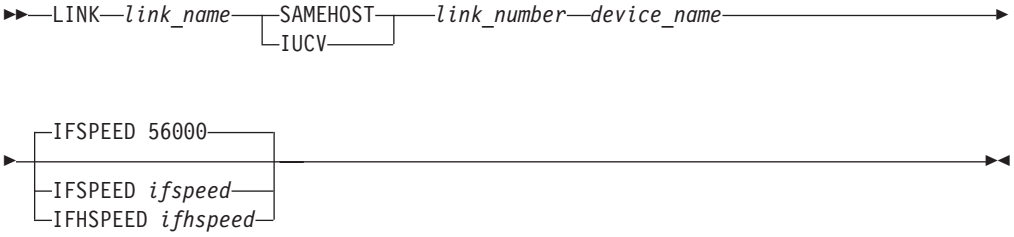
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCPIP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 8 characters. The same name is specified in the SNALINK LU6.2 configuration data set (*hlq*.PROFILE.TCPIP) to identify this link.

SAMEHOST
A constant that specifies that the device for SNA LU type 6.2 support uses a SAMEHOST connection.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

link_number
Must be an integer, but the 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 same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can specify multiple LU6.2 DEVICE statements for the same started procedure. A single LU6.2 application can support multiple SAMEHOST links. A SAMEHOST link is created for each pair of LU6.2 DEVICE and LINK statements.

You would then need to configure multiple LINK statements in the LU6.2 configuration data set using names that correspond to those in the *hlq*.PROFILE.TCPIP LINK statements.

You can add new DEVICE and LINK statements using the VARY TCPIP command but cannot modify any existing ones.

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 OBEYFILE data set.

Usage Notes

IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.

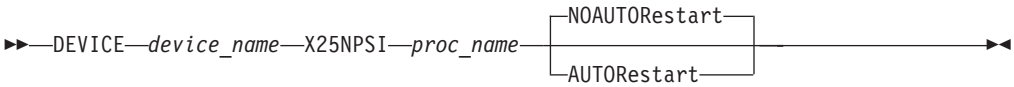
Related Topics

"VARY Command—TCPIP Address Space" on page 264

DEVICE and LINK Statement—X.25 NPSI Connections

Use the DEVICE statement to specify the name and address of the X.25 NPSI interface program devices that you use. Use the LINK statement to define a network interface link associated with the X.25 NPSI interface program devices.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

X25NPSI
Specifies that the device is an X.25 NPSI.

proc_name
The name of the X.25 NPSI server started procedure.

AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

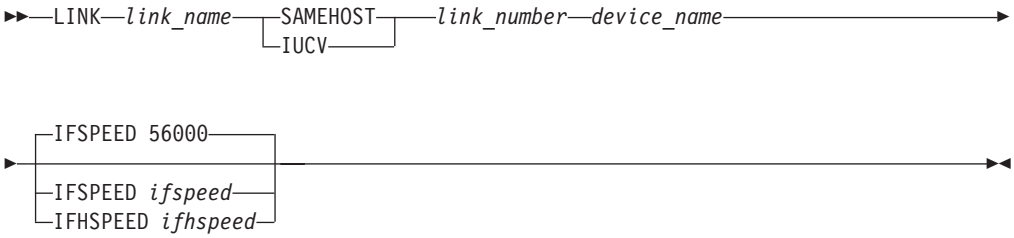
Note:

TCPIP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Note: Only one DEVICE and LINK statement per TCPIPX25 address space is allowed.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

SAMEHOST
Specifies that the connection to X.25 NPSI is established using a SAMEHOST connection.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

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 same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command but cannot modify any existing ones.

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 OBEYFILE data set.

Examples

This example shows how you might code DEVICE, LINK, and related statements for an X.25 connection.

```
DEVICE X25DEV X25NPSI TCPIPX25
LINK X25LINK SAMEHOST 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; Network   First Hop   Link name   Packet size   Subnet mask   Subnet Value
  192.005   =           X25LINK     2000          0.0.255.0     0.0.58.0
;
START X25DEV
;
```

Usage Notes

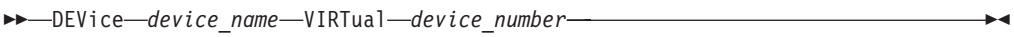
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- To use dynamic routing with this device, see “Configuring NBMA Subnetworks” on page 1029.

DEVICE and LINK Statement—Virtual Devices

Use the DEVICE statement to specify a device name and virtual number of a virtual device.

Use the LINK statement to define the link on the DEVICE statement.

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

VIRTUAL
Specifies that the device is not associated with real hardware and is used for fault tolerance support. The virtual devices always stay active and are never subject to physical failure.

device_number
Must be a hexadecimal number, but the value is ignored. This parameter is included for consistency with the DEVICE statements for other device types.

Syntax

Only one LINK statement can be defined for each virtual device.

Parameters

link_name
The name of the link. The maximum length is 16 characters. The same name is specified in the HOME statement.

VIRTUAL
Specifies that the link is a virtual link that is not associated with real hardware and is used for fault tolerance support.

adapter_address
Must be an integer, but the value is ignored. This parameter is included for consistency with the LINK statements for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE

- DELETE linkname and DELETE devicename statements

3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command.

When you add new LINK statements, any corresponding GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the GATEWAY, HOME, or TRANSLATE statements in the OBEYFILE data set, be sure to include existing links that you want to have active in your configuration.

Examples

```
DEVICE VDEV1  VIRTUAL 0
LINK   VLINK1 VIRTUAL 0 VDEV1
DEVICE VDEV2  VIRTUAL 1
LINK   VLINK2 VIRTUAL 0 VDEV2
```

Usage Notes

- Only one virtual link can be defined for a virtual device.
- More than one virtual DEVICE/LINK statement can be defined to allow for multiple virtual IP addresses on one TCP/IP image in one MVS system.
- A virtual LINK cannot be coded on the START, GATEWAY or TRANSLATE statements, but can be coded on a BSDROUTINGPARMS statement for interface characteristics such as subnet mask.
- For rules on defining virtual IP addresses for virtual links, see the HOME statement on page “HOME Statement” on page 207.
- Because a virtual device cannot be started or stopped and is always active, it cannot be deleted.
- If you are using a nameserver to resolve host names and any of the related resolver configuration files have only one NSINTERADDR statement (IP address of a name server) coded that specifies a VIPA address, then the host that the nameserver is running on must be configured to use SOURCEVIPA.
- If you are running with 3172s configured for multi-host connectivity (release 3.5 and above) and wish to use VIPA addresses on the host, then you must configure the 3172 as one of the following:
  - As a default router (will route all IP addresses)
  - Configure all VIPA addresses in the 3172

Related Topics

- “BSDROUTINGPARMS Statement” on page 140
- “HOME Statement” on page 207

DEVICE and LINK Statement—3745/46 Channel DLC Devices

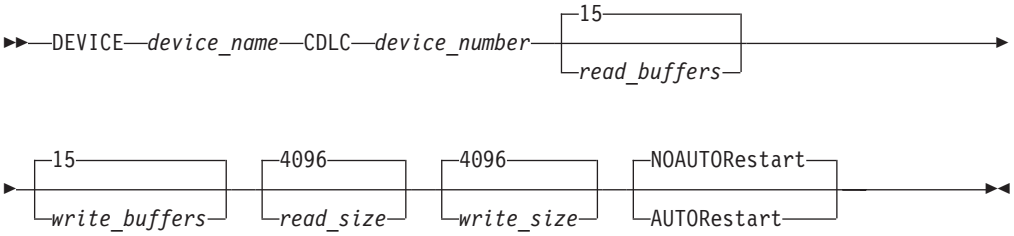
Use the DEVICE statement to specify the name and hexadecimal device number of the channel data link control (CDLC) devices that you use. Use the LINK statement to define a network interface link associated with the CDLC devices.

If the device is running NCP V7R3 and dynamic routing is to be performed, SNALINK must be configured to carry RIP transport PDUs:

- NCP V7R3 does not support native IP transmission across the channel of the transport PDUs associated with RIP traffic (NCP V7R3 expects these PDUs to be carried in SNA Frames). SNALINK is still required in environments where dynamic routing is performed with the NCP V7R3 (via NCPROUTE).

  To minimize the amount of data sent across the SNALINK (LUO) connection (as SNALINK consumes more CPU than does IP over CDLC), it is recommended that the RIP Filter be used to cause RIP updates to be sent across the channel, while the associated transport PDUs (Route Table Management, for example, Handshaking, Add Route Request, Delete Route Request) are carried over the SNALINK connection.
- If the device is running NCP V7R3 or above, or if the device is a 3746 model 950, SNALINK is not required (all IP and RIP traffic can be transported over TCP/IP's direct CDLC link).

Syntax



Parameters

device_name
The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CDLC
Specifies that this device is to run the CDLC protocol.

device_number
The hexadecimal device number of the CDLC device.

read_buffers
The decimal number of buffers to allocate to the read channel program. The default is 15. The minimum is 1 and the maximum is 63. The product of read_buffers times read_size must be less than or equal to 65535. If the product of these configured variables exceeds 65535, TCP/IP will reduce read_buffers to the integer 65535/read_size.

write_buffers
The decimal number of buffers to allocate to the write channel program. The minimum is 1 and the maximum is 63. The product of write_buffers times

write_size must be less than or equal to 65535. If the product of these configured variables exceeds 65535, TCP/IP will reduce write_buffers to the integer 65535/write_size.

read_size
The size in bytes (decimal) of the read buffers. The default is 4096. Valid values are 1024, 2048, and 4096.

write_size
The size in bytes (decimal) of the write buffers. The default is 4096. Valid values are 1024, 2048, and 4096.

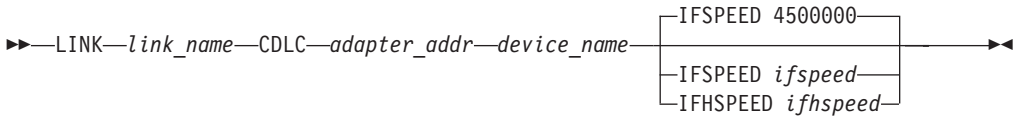
AUTORESTART
In the event of a device failure, the TCP/IP address space will attempt to reactivate the device.

Note:

TCP/IP automatically attempts reactivation of the device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation three times at 30 second intervals following *all* device-failure indications.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space will not attempt to reactivate this device.

Syntax



Parameters

link_name
The name of the link. The maximum length is 16 characters.

CDLC
Specifies that the link is a channel DLC.

adapter_addr
Must be an integer, but the value is ignored. This parameter is included for consistency with the LINK statement formats for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

Modifying

To modify any DEVICE and LINK statement values, follow these steps:

1. Stop the device.
2. Use a VARY TCP/IP command with an OBEYFILE that contains:
   - A new HOME statement which does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements
3. Use a VARY TCPIP command with an OBEYFILE that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added
4. Start the device.

Note: To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in the above list.

See "VARY Command—TCPIP Address Space" on page 264 for more information about the VARY TCPIP commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command but cannot modify any existing ones.

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 OBEYFILE data set.

Usage Notes

- A maximum of one link statement can be defined for each CDLC device.
- IFSPEED and IFHSPEED are mutually exclusive. If either one is specified, both SNMP MIB objects are set. If IFSPEED is specified, IFHSPEED is IFSPEED/1000000. If IFHSPEED is specified, IFSPEED is the minimum of IFHSPEED*1000000 or the maximum IFSPEED value.
- For a buffer size of 4096, the maximum number of buffers is 15. For a buffer size of 2048, the maximum number of buffers is 31. For a buffer size of 1024, the maximum number of buffers is 63.

Related Topics

"VARY Command—TCPIP Address Space" on page 264

GATEWAY Statement

Use the GATEWAY statement to add static routes to the IP route table.

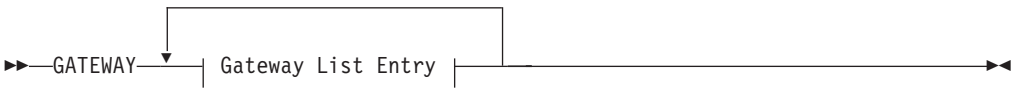
The IP route table can be modified by replacing the table using the VARY TCPIP command, by incoming ICMP redirect packets sent from adjacent machines, or by updates from dynamic routing services (OMPROUTE, OROUTED).

The first GATEWAY statement of each configuration data set executed replaces the existing routing table with the new gateway information. All static routes are deleted, but routes created by OMPROUTE or OROUTED are not deleted. Subsequent GATEWAY statements in the same data set add entries to the routing table.

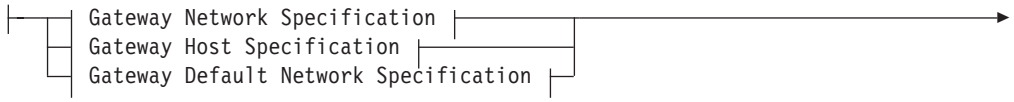
Notes:

1. If OMPROUTE or OROUTED is running, static routes defined by the GATEWAY statement cannot be deleted by OMPROUTE or OROUTED. If you want OMPROUTE or OROUTED to manage all routes, an empty GATEWAY statement can be used to eliminate the static routes. OMPROUTE or OROUTED will find out about them dynamically.
2. VIPA links are not allowed on the GATEWAY statement.
3. When an incorrect GATEWAY statement entry is encountered, if the remaining entries on that GATEWAY statement can be syntactically checked, they are processed. Otherwise, they are ignored. Subsequent GATEWAY statements in the same profile or obeyfile are processed.
4. A specific host route takes precedence over a subnetwork route, followed by a network route, followed by a supernetwork route, and finally, a default route.

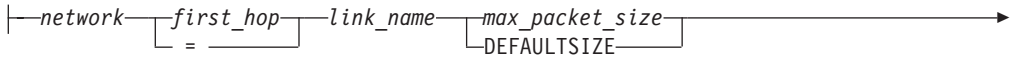
Syntax



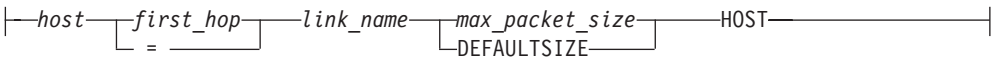
Gateway List Entry:



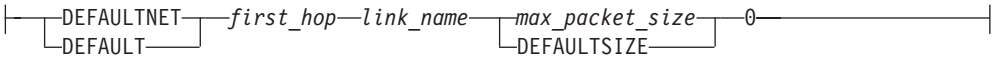
Gateway Network Specification:



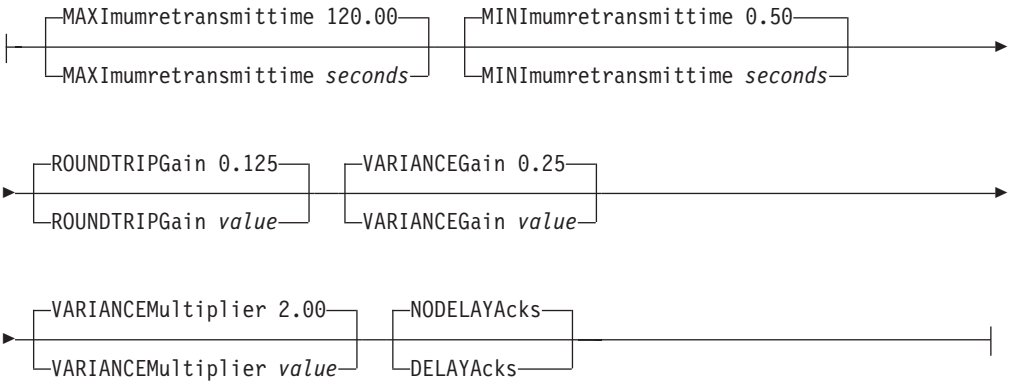
Gateway Host Specification:



Gateway Default Network Specification:



Gateway List Entry Options:



Note: Use the Netstat *CONFIG* or *-f* option to determine the actual default value. Use the Netstat *GATE* or *-g DETAIL* option to determine the actual value used for a specific GATEWAY.

Parameters

network

The IP address in dotted-decimal form.

- An example of a class A network is 9.0.0.0.
- An example of a class B network is 129.34.0.0.
- An example of a class C network is 192.9.100.0.

Use the *subnet_mask* and *subnet_value* fields to define a subnetted network.

DEFAULTNET

Specifies the default to use for any network not explicitly routed. DEFAULTNET can be specified only once.

For the DEFAULTNET keyword to take effect, you must specify a GATEWAY statement that defines the route to the *first_hop*.

DEFAULT

Multiple DEFAULT entries can be specified, allowing for multiple default routes in addition to the DEFAULTNET route. If the DEFAULTNET route does not exist or is not active (that is, the link it is defined to is not active), the first active DEFAULT route is used when no specific route matches the destination or source IP address.

host
The host address, specified as 4 octets (192.9.100.3, for example). If a host address is specified, the keyword HOST must be specified in place of the *subnet_mask* field, and the *subnet_value* field must not be specified.

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. This is not supported for DEFAULTNET or DEFAULT.
- The internet address of a gateway or router that you can reach directly, and that forwards messages for the destination network or host.

link_name
The name of the link through which packets are sent to the specified network. The link name is defined in a LINK statement.

max_packet_size
The maximum transmission unit (MTU) in bytes for the network or host. This value can be up to 65535.

The special entry DEFAULTSIZE in this field requests that TCP/IP supply a default value of 576. We recommend you use the following sizes instead of DEFAULTSIZE as the packet size for these networks:
1492 bytes for Ethernet 802.3
1500 bytes for Ethernet Version 2 IEEE
1500, or 2000 or 4000 bytes for token ring
4352, or 2000 or 4000 bytes for FDDI
65527 bytes for CTC
4096 bytes for CLAW

You can change these values as needed later during performance tuning.

subnet_mask
A bit mask (expressed in dotted-decimal form) having bits in the network and/or host portions that defines the subnet mask associated with the route. If a route to the network is not subnetted, specify a subnet_mask of zero and omit the subnet_value. For a specific host route, specify a subnet_mask of HOST and omit the subnet_value. For subnetted and supernetted routes, the subnet mask can be in the several forms depending upon the VARSUBNETTING setting in the ASSORTEDPARMS or IPCONFIG statement as follows:

- NOVARSUBNETTING

  The network portion is coded as zeroes and the host portion shows the bits that make up the subnet. The bits must be contiguous from left to right. For a subnet route, specify the subnet_mask in the host portion and specify the subnet_value. Single, fixed-length subnet masks must be defined for all subnet routes to a common network, that is, multiple subnets having the same network number must have identical subnet masks.
- VARSUBNETTING

  Variable-length subnet masks may be used in a single network, that is, multiple subnets having the same network number can have different subnet masks. For a subnet route, specify the network portion as zeroes, the subnet_mask in the host portion, and the subnet_value. The bits in the subnet mask must be contiguous from left to right.

  For a supernet route, which can be used to represent multiple network routes, specify a supernet mask and omit the subnet_value. The supernet mask must be coded such that in the network portion the supernet portion is

zero and the remainder is ones to represent the multiple networks. Also, the host field must be zero. The supernet mask is calculated by taking the difference of the network class mask and the bit-contiguous supernet mask in BSD form.

For example, for a supernet 130.200 having the supernet mask of 255.252.0.0, the difference of the network class B mask 255.255.0.0 and the supernet mask of 255.252.0.0 yields the supernet mask of 0.3.0.0. Based upon the unmasked bits in the network portion, the supernet 130.200 represents networks 130.200, 130.201, 130.202, and 130.203.

Note: The mask bits of all ones in the host portion cannot be used for the subnet mask.

subnet_value
Value of each *subnet_mask*. Each subnet should have a unique dotted-decimal representation. A value of 0 indicates the default route for any subnet of this network that is not specifically routed. Do not include the *subnet_value* field if the *subnet_mask* is 0, HOST, or contains a supernet mask.

If the network has one or more subnets, specify a separate entry in the GATEWAY statement for each subnet. The network part of each GATEWAY entry is identical (contains the IP network address as if the network has no subnets). The *subnet_mask* part of each GATEWAY entry is also identical, but the *subnet_value* varies.

MAXIMUMRETRANSMITTIME
Used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet.

MINIMUMRETRANSMITTIME
Used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet.

ROUNDTRIPGAIN
Used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet.

VARIANCEGAIN
Used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet.

VARIANCEMULTIPLIER
Used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet.

DELAYACKS
Delays transmission of acknowledgements when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the GATEWAY statement only applies to the TCP protocol and only affects acknowledgements returned to the foreign host. To delay acknowledgements on a TCP/IP port connection, use the DELAYACKS parameter on the PORT statement.

NODELAYACKS
Specifies that an acknowledgement is returned immediately. This is the default.

Modifying

To modify any values on the GATEWAY statement, use a VARY TCPIP command with an OBEYFILE which contains a new GATEWAY statement. All existing static routes will be deleted. To remove all static routes from the IP routing table, specify

an empty GATEWAY statement. See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

Notes:

1. If any HOME statement values were dynamically changed, all static routes that correspond with the LINK names in the GATEWAY statements will be deleted.
2. If any DEVICE/LINK statement values were dynamically deleted, all static routes that correspond with the LINK names on the GATEWAY statement will be deleted. In the OBEYFILE containing changed DEVICE/LINK statements and a new HOME statement, include a new GATEWAY statement for the VARY TCPIP command.
3. If a VARY TCPIP command was issued to stop a device, all static routes in the GATEWAY statement that correspond with the LINK names belonging to that device will be inactivated.
4. If a VARY TCPIP command was issued to start a device, all static routes in the GATEWAY statement that correspond with the LINK names belonging to that device will be activated.
5. Because the IP routing tables can include both static and dynamic routes, the dynamic routes are not affected by GATEWAY statement changes. The dynamic routes are managed by the dynamic routing protocols.

Examples

- Following is an example of a network and the corresponding GATEWAY statements.
- This example shows a GATEWAY statement that is divided by the types of routes used.

```
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network  First hop  Link name Packet size  Subnet mask  Subnet value
  193.9.200          =        LINK1  DEFAULTSIZE  0
  193.0.2            =        LINK2  DEFAULTSIZE  0

 ;
; Indirect Routes - Routes that are reachable through routers on my
;                   network.
;
; Network  First hop  Link name Packet size  Subnet mask  Subnet value

  128.84        193.9.200.2   LINK1  DEFAULTSIZE  0.0.255.0   0.0.1.0
  128.84        193.9.200.100 LINK1  DEFAULTSIZE  0.0.255.0   0.0.55.0
  128.84        193.0.2.2     LINK2  DEFAULTSIZE  0.0.255.0   0
  9.67.43.126   193.0.2.2     LINK2  DEFAULTSIZE  HOST
;
;
; Default Route - All packets to an unknown destination are routed
;                 through this route.
;
; Network  First hop  Link name Packet size  Subnet mask  Subnet value

  DEFAULTNET    193.0.2.3     LINK2  DEFAULTSIZE  0
```

- To use an ATM SVC link as a virtual link, be sure to define the virtual link in the correct subnet. Below, two systems, six and seven, have an ATM SVC link between them. The definition in the GATEWAY statement for the ATM SVC link and the virtual link associated with it are shown. The address of the end point of the ATM SVC link is 213.1.5.7 and the virtual link is to 213.1.1.2.

```
213.1.5         =          SATM6TO7  8192       0
213.1.1.2  213.1.5.7       SATM6TO7  8192       HOST
```

Assuming an ATM PVC link is defined at 213.1.4.7, the GATEWAY statements would be:

```
213.1.4         =          PATM6TO7  8192       0
213.1.1.2  213.1.4.7       PATM6TO7  8192       HOST
```

Figure 10 on page 199 shows a host, MVS1, directly connected to variable subnets 10.2 and 10.2.8. MVS1 is indirectly connected to variable subnets 121.2.3.128 and 121.2.4. In addition, MVS1 is directly connected to supernet 192.3.200 and indirectly connected to supernet 130.200.

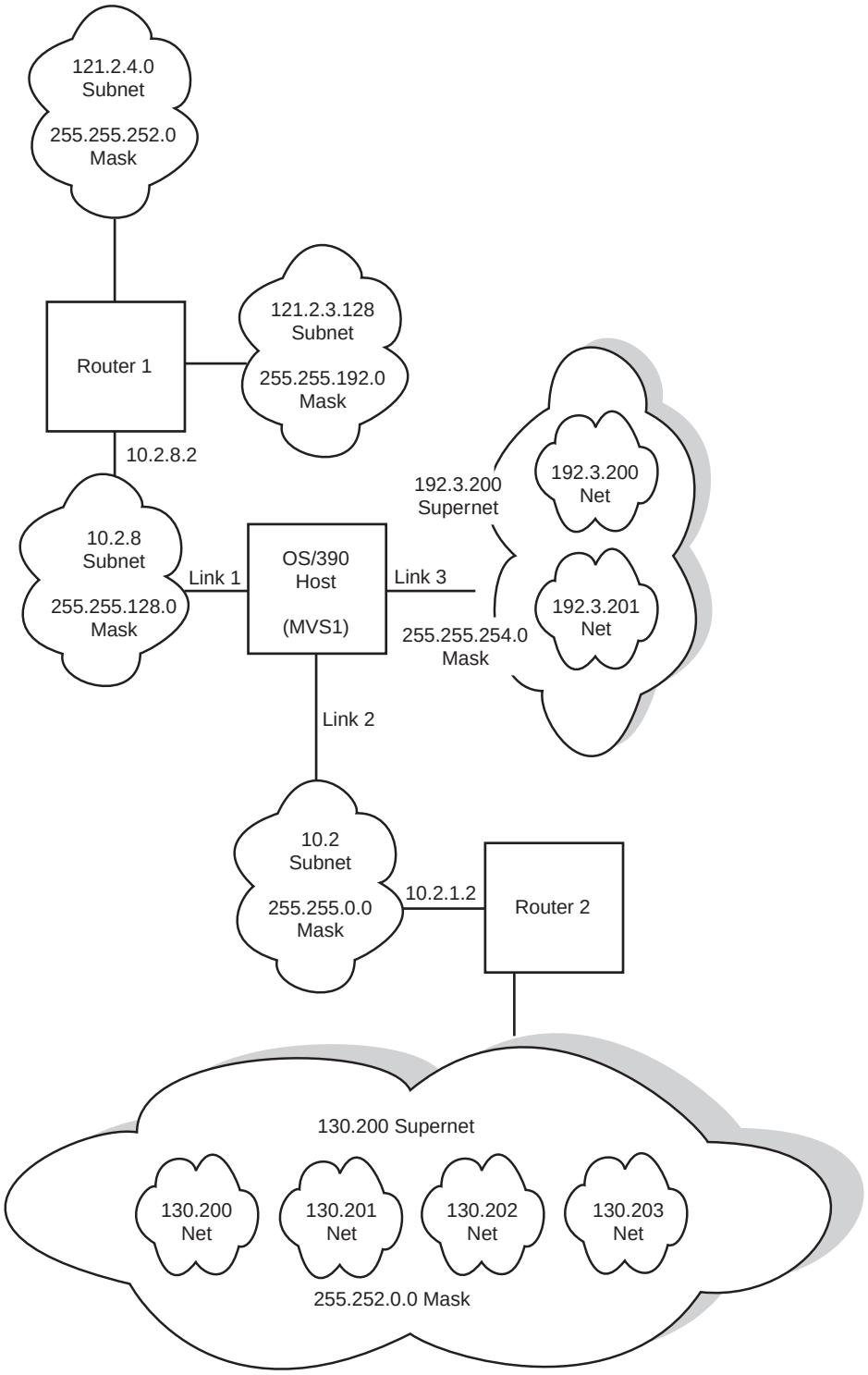


Figure 10. Example of Network Connectivity Using Variable Subnetting

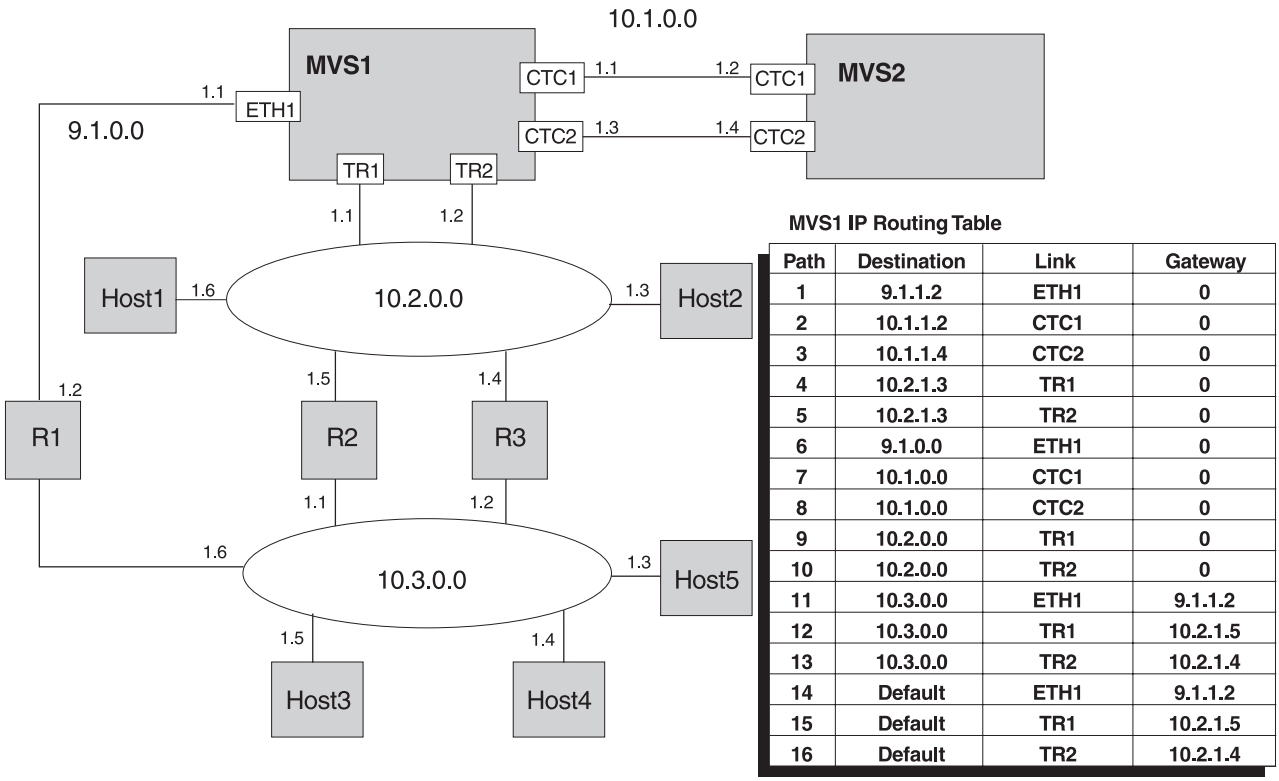
The following is an example of the corresponding GATEWAY statement:

```
GATEWAY
;
; Direct and indirect subnets
```

```
;
; net_number first_hop link  pkt_sz subnet_mask  subnet_value
  10            =      LINK1  1500  0.255.128.0  0.2.8.0
  10            =      LINK2  2000  0.255.0.0    0.2.0.0
  121.2      10.2.8.2  LINK1  2000  0.0.255.192  0.0.3.128
  121.2      10.2.8.2  LINK1  2000  0.0.252.0    0.0.4.0
;
; Direct and indirect supernets
;
  192.3.200     =      LINK3  2000  0.0.1.0      0
  130.200    10.2.1.2  LINK2  2000  0.3.0.0      0
```

Network 10 has variable subnets 10.2 and 10.2.8. Network 121.2 has variable subnets 121.2.3.128 and 121.2.4. Subnet 10.2 is using the high-order byte of the host field as the subnet field and subnet 10.2.8 is using the first two bytes of the host field as the subnet fields. Networks 130.200 and 192.3.200 are supernets representing multiple networks and uses the high-order bytes of the network field as the supernet fields. The number of networks is determined by the mask bits in a network field. For example, the subnet masks of 0.0.1.0 and 0.3.0.0 are interpreted as supernet masks of 255.255.254.0 and 255.252.0.0 respectively. They are calculated by taking the differences of the network class masks and the supernet masks. For example, the difference of the network class B mask of 255.255.0.0 and the supernet mask of 255.252.0.0 yields the subnet mask of 0.3.0.0. Based upon the masked bits in the network field, supernet 192.3.200 represents networks 192.3.200 and 192.3.201. Supernet 130.200 represents networks 130.200, 130.201, 130.202, and 130.203. Note that the subnet value is zero for the supernets.

Figure 11 on page 201 shows equal-cost multipath routes to common destinations defined in MVS1 to be used for IP traffic load splitting. Paths 4 and 5 illustrate multiple equal-cost direct host routes. Paths 7, 8, 9, and 10 illustrate multiple equal-cost direct subnetwork routes. Paths 11, 12, and 13 illustrate multiple equal-cost indirect subnetwork routes. Paths 14, 15, and 16 illustrate multiple equal-cost default routes.



MVS1 IP Routing Table

Path	Destination	Link	Gateway
1	9.1.1.2	ETH1	0
2	10.1.1.2	CTC1	0
3	10.1.1.4	CTC2	0
4	10.2.1.3	TR1	0
5	10.2.1.3	TR2	0
6	9.1.0.0	ETH1	0
7	10.1.0.0	CTC1	0
8	10.1.0.0	CTC2	0
9	10.2.0.0	TR1	0
10	10.2.0.0	TR2	0
11	10.3.0.0	ETH1	9.1.1.2
12	10.3.0.0	TR1	10.2.1.5
13	10.3.0.0	TR2	10.2.1.4
14	Default	ETH1	9.1.1.2
15	Default	TR1	10.2.1.5
16	Default	TR2	10.2.1.4

Figure 11. Example of Equal-cost Multipath Routes

The following is an example of the corresponding GATEWAY statement.

```
GATEWAY
;
; Direct host routes
;
; Network   First hop   Link name  Packet size  Subnet mask Subnet value
   9.1.1.2   =            ETH1        1500        HOST
   10.1.1.2  =            CTC1        65527       HOST
   10.1.1.4  =            CTC2        65527       HOST
   10.2.1.3  =            TR1         4000        HOST
   10.2.1.3  =            TR2         4000        HOST
;
; Direct subnet  routes
;
; Network   First hop   Link name  Packet size  Subnet mask Subnet value
   9         =            ETH1        1500        0.255.0.0   0.1.0.0
   10        =            CTC1        65527       0.255.0.0   0.1.0.0
   10        =            CTC2        65527       0.255.0.0   0.1.0.0
   10        =            TR1         4000        0.255.0.0   0.2.0.0
   10        =            TR2         4000        0.255.0.0   0.2.0.0
;
; Indirect subnet  routes
;
;  Network  First hop   Link name  Packet size  Subnet mask Subnet value
   10        9.1.1.2     ETH1        1500        0.255.0.0  0.3.0.0
   10        10.2.1.5    TR1         1500        0.255.0.0  0.3.0.0
   10        10.2.1.4    TR2         1500        0.255.0.0  0.3.0.0
;
;  Default routes
;
;  Network  First hop   Link name  Packet size  Subnet mask Subnet value
   DEFAULT   9.1.1.2     ETH1        1500        0
   DEFAULT   10.2.1.5    TR1         1500        0
   DEFAULT   10.2.1.4    TR2         1500        0
```

Note: The DEFAULTNET parameter cannot be used for defining multiple default route entries.

Usage Notes

- For an MPCPTP link with multiple point to point connections, a first hop address must be specified for outbound multicast datagrams on a GATEWAY statement. If no GATEWAY statement is specified with a first hop address, outbound multicast datagrams will be silently discarded.
- Packet size considerations:
  - A network IP address is distinguished from a host IP address by determining if the host portion of the IP address is all zeroes as determined by the IP address class:
    - Class A — *a*.0.0.0 where 0<=*a*<=127
    - Class B — *a.b*.0.0 where 128<=*a*<=191, 0<=*b*<=255
    - Class C — *a.b.c*.0 where 192<=*a*<=223, 0<=*b*<=255, 0<=*c*<=255

    Therefore, it is not valid to have a host (or source or destination IP address) defined at an internet address containing a zero for the host (for example, 127.0.0.0).
  - 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 consumes a fixed amount of CPU time for each packet, independent of the packet size.
  - The *max_packet_size* that CS for OS/390 can handle varies for different networks. 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 bytes.
  - 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.
    - The TCP maximum segment size for the 3172 Interconnect Controller Program is 4096. Any packet specifications over 4096 are limited by this restriction. For example, if you specified a packet size of 4352, the resulting packet size would still only be 4096 + the header = 4132.
  - Large packets can be fragmented by intervening gateways. Fragmentation and reassembly of packets are expensive in their use of bandwidth and CPU time. Therefore, 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.
  - If this is a RISC System/6000 link, then the *max_packet_size* cannot exceed the *write_size* specified on the corresponding DEVICE statement.
  - Occasionally, your packets will pass through routers that fragment packets to the internet default size (576 bytes). Use the GATEWAY configuration statement to further reduce packet sizes. For example, the router to network 192.8.4 is reached through router 14.0.0.10., and somewhere along the path, packets larger than 460 bytes are fragmented. Throughput can be improved using the following GATEWAY statement:

```
GATEWAY
; Network   first-hop   link   packet-size    subnet-mask
  192.8.4   14.0.0.10   LINK1     460            0
```

- Considerations for retransmission parameters:

The re-transmission parameters allow system administrators who are familiar with TCP/IP transmission performance to alter the flow of TCP/IP data packets and acknowledgements. Normally,

– TCP typically waits to receive two packets before sending one ACK to acknowledge the data within them.
– When TCP sends a packet, it waits for an acknowledgement. If it times out before getting an acknowledgement, it resends the packet.

With the following parameters you can adjust the re-transmission time-out calculations to allow for slower transmission so that packets are not resent as quickly.

MAXIMUMRETRANSMITTIME
MINIMUMRETRANSMITTIME
ROUNDTRIPGAIN
VARIANCEGAIN
VARIANCEMULTIPLIER

TCP uses these values in an algorithm called the *TCP Re-transmission Timeout Calculation*. As a result of this calculation, TCP/IP:

– Uses the minimum round-trip time as long as it is less than the MINIMUMRETRANSMITTIME.
– Uses the average round-trip time if round-trip time is greater than or equal to the MINIMUMRETRANSMITTIME.
– Uses the maximum round-trip time if round-trip time is greater than the MAXIMUMRETRANSMITTIME.

You can specify ROUNDTRIPGAIN, VARIANCEGAIN, and VARIANCEMULTIPLIER to tell TCP how heavily it should weight the most recent behavior of the network versus the long term behavior. If you specify smaller values for these parameters, then TCP will attempt to correct for congestion only if the congestion is sustained. With larger values, TCP will correct for congestion more quickly and the system will be more sensitive to variations in network performance. We recommend using the default values unless you find your re-transmission rate is too high.

With DELAYACKS, you can delay the acknowledgements so that they can be combined with data sent to the foreign host.

- If the gateway table is empty, the LOOPBACK test address is still routed properly. For information on testing commands with LOOPBACK, see the *OS/390 SecureWay Communications Server: IP User's Guide*.
- If IPCONFIG VARSUBNETTING is specified, the 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 0.255.254.0 is valid (an actual mask of 255.255.254.0 or FFFFE00), but a class A mask of 0.255.253.0 is not valid (an actual mask of 255.255.253.0 or FFFFD00) because 253 is 11111101.
- Static indirect routes can not be added to the routing table if the device the route is going over is not started when the gateway's stanza is processed. To ensure that the indirect static routes are processed as expected, code a direct route as the indirect routes as follows:

```
GATEWAY
9                =          U42TR          4096   0.255.192.0 0.67.0.0
9.67.128.87   9.67.17.10    U42TR          4096       HOST
9.67.128.117  9.67.17.10    U42TR          4096       HOST
9.67.134.212  9.67.17.10    U42TR          4096       HOST
```

The static direct route in allows the stack to process the indirect routes during initialization.

- There is no limit on the number of equal-cost multipath routes to a destination network or host.

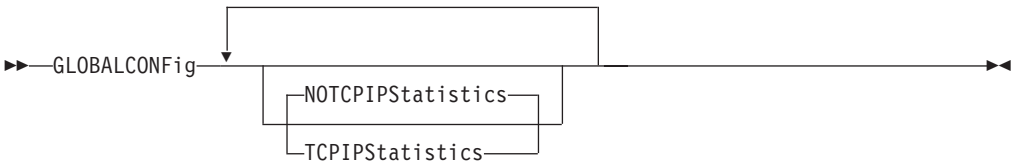
Related Topics

- “DEVICE and LINK Statements” on page 101
- “VARY Command—TCPIP Address Space” on page 264
- “BSDROUTINGPARMS Statement” on page 140

GLOBALCONFIG Statement

Use the GLOBALCONFIG statement to pass global configuration parameters to TCP/IP.

Syntax



Parameters

TCPIPSTATISTICS
Prints the values of several TCP/IP counters to the output data set designated by the CFGPRINT JCL statement. These counters include number of TCP retransmissions and the total number of TCP segments sent from the MVS TCP/IP system.

The TCPIPSTATISTICS parameter is confirmed by the message:

```
TCPIPSTATISTICS is enabled
```

Note: This parameter is identical to the ASSORTEDPARMS TCPIPSTATISTICS parameter (see "ASSORTEDPARMS Statement" on page 128). Both parameters print identical results in the Global Configuration Information section.

However, the SMFCONFIG TCPIPSTATISTICS parameter (see "SMFCONFIG Statement" on page 239) serves a different purpose. Requests that SMF records of subtype 5 containing TCP/IP statistics are created. Note that these records are created periodically based on the SMF interval in effect.

NOTCPIPSTATISTICS
Indicates that the values of statistical counters should not be printed in the Global Configuration Information section of the output for the netstat (or onestat) command.

The NOTCPIPSTATISTICS parameter is confirmed by the message:

```
TCPIPSTATISTICS is disabled
```

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example shows the use of the NOTCPIPSTATISTICS parameter on the GLOBALCONFIG statement.

```
GLOBALCONFIG NOTCPIPSTATISTICS
```

Usage Notes

If TCPIPSTATISTICS is also specified on an ASSORTEDPARMS statement, the setting from the last statement processed is used. For example, if NOTCPIPSTATISTICS is specified on a GLOBALCONFIG statement, but TCPIPSTATISTICS is specified on a subsequent ASSORTEDPARMS statement, the values of the statistical TCP/IP counters will be printed to the output data set specified by the CFGPRINT JCL statement.

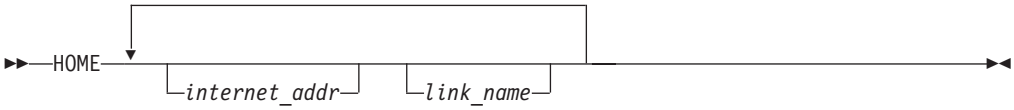
Related Topics

- "ASSORTEDPARMS Statement" on page 128

HOME Statement

The HOME statement provides the list of home addresses and associated link names (called the HOME list).

Syntax



Parameters

internet_addr
The internet address valid for this host. The internet address can be associated with a physical or VIPA link. The internet address must be specified in dotted-decimal form.

link_name
The name of the link defined in a previous LINK statement (or the reserved name LOOPBACK) that is associated with the home address.

Modifying

To modify the HOME statement, use a VARY TCPIP command with an OBEYFILE that defines a new HOME statement.

Notes:

1. If you use the HOME statement to change the IP addresses of any links, you should stop and restart the affected devices.
2. The first HOME statement of each configuration data set executed replaces the existing HOME list with the new list; subsequent HOME statements in the same data set add entries to the list. However, dynamically-defined HOME list entries created by XCF dynamics are not deleted by a new HOME statement.

Usage Notes

- A HOME address used by an ATM LINK referencing an ATMLIS should be within the logical IP subnetwork defined by the LIS subnet_value and subnet_mask. If it is not within the subnetwork, the link cannot be used for sending or receiving any ATM SVC traffic.
- Only one home address can be associated with a link. If the same link is specified in more than one HOME list entry, only the home address in the last entry is associated with the link. The only exception to this is the LOOPBACK link. Multiple home entries are accepted for LOOPBACK in addition to 127.0.0.1.

  A *link_name* of LOOPBACK defines the IP address to use for LOOPBACK. No DEVICE or LINK statement is needed for LOOPBACK, and it cannot be started or stopped (LOOPBACK is always active).

  You can use LOOPBACK in the HOME list only to define additional LOOPBACK addresses. If you try to redefine the default LOOPBACK address of 127.0.0.1, it is flagged as a duplicate entry.
- The primary interface IP address is used as the source IP 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. A default local address is used for the GETHOSTID() function.

- If the first HOME statement of a profile contains no internet address or link name, all addresses are removed from the HOME list except for the loopback address.
- When an incorrect HOME entry is encountered, all entries following that entry on that HOME statement are ignored. Subsequent HOME statements are processed.
- When defining VIPA addresses, observe the following rules and recommendations:
  - It is recommended that a primary VIPA address be coded as first in the HOME list or be coded on the PRIMARYINTERFACE statement to serve as the default local address for usage by the GETHOSTID() function.

    In general, the VIPA addresses can be coded in any order in the HOME list; however, if you specify SOURCEVIPA on the ASSORTEDPARMS or IPCONFIG statement, the order of the VIPA addresses are important in terms of how source IP addresses are used for outbound datagrams originating at the host. In this case, observe the following rules:
    - In the HOME list, the VIPA address that closely precedes a physical IP address will be used as the source IP address.
    - If VIPA addresses are coded after all of the physical IP addresses, no VIPA addresses will be used as the source IP address.

    Note: See the Examples section for more information on configuring SOURCEVIPA addresses.
  - A VIPA address must be a unique host address in the network and not be a duplicate of any physical IP address in the network.
  - If using RIP services (OMPROUTE, OROUTED) and Host Routing Broadcasting is not supported by adjacent routers (that is, inability to learn host routes), the following restrictions for VIPA addresses must be applied in order to benefit from fault tolerance support:
    - If you use subnetting and VIPA addresses are in the same network as the physical IP addresses, the subnetwork portion of any VIPA addresses must not be the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the VIPA address.
    - If subnetting is not used on any physical interface, the network portion of any VIPA addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the VIPA address, preferably a class C network address.
  - If using RIP services (OMPROUTE, OROUTED) and adjacent routers support receiving host routes from the RIP responses for routing updates, the network or subnetwork portions of VIPA addresses can be the same across multiple OS/390 TCP/IP stacks in the network. See the OMPROUTE or OROUTED topics for more information about how to include the optional host routes in the RIP responses for routing updates on the RIP services.
  - More than one VIPA address can be defined in one network or subnetwork.
  - You can use the VIPA address as the primary or only destination for the name of an OS/390 server on the domain name server. A workstation on the network would use the OS/390 server name (translated into the VIPA address) to access applications on the OS/390 server.
- While a VIPA address can be assigned to each TCP/IP stack in one OS/390 image, it is recommended that an internal point-to-point link (for example, CTC)

be defined between the stacks. This ensures that the VIPA address in one OS/390 TCP/IP stack attached to a failing adapter/controller (for example, 3172) can be reached via another OS/390 TCP/IP stack channel-attached to the same controller through another adapter or to another controller across the point-to-point link.

- For more information on what routing protocols to use to achieve non-disruptive TCP-connection fault tolerance, see “Chapter 3. Virtual IP Addressing” on page 71.
- 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 VIPA address, the host the nameserver is running on must be configured to use SOURCEVIPA.

Examples

This example shows a HOME statement that defines the IP addresses of each link in the host.

```
HOME
   151.4.1.2      TR2
   192.1.1.1      VIPA1
   130.50.75.1    TR1
   193.5.2.1      ETH1
   192.2.1.1      VIPA2
   9.67.43.110    FDDI1
   193.7.2.1      SNA1
```

VIPA1 and VIPA2 are examples of VIPA links associated with VIPA addresses, while others are examples of physical links associated with physical IP addresses. If you specify SOURCEVIPA on the IPCONFIG or ASSORTEDPARMS statement, VIPA1 serves as the VIPA address for TR1 and ETH1, and VIPA2 for links FDDI1 and SNA1. TR2, because there is no VIPA definition preceding it in the HOME list, is not affected by SOURCEVIPA. The VIPA addresses are used in the outbound IP datagrams. For more information, see “IPCONFIG Statement” on page 213 and “ASSORTEDPARMS Statement” on page 128.

The following example shows the definition of an additional LOOPBACK address:

```
HOME  9.67.113.105   CTCD00   ; CTC IP address for this system
      14.0.0.0       LOOPBACK ; additional LOOPBACK address
```

If using the SOURCEVIPA option for the outbound datagrams originating at an OS/390 TCP/IP stack, see the following example.

Select a VIPA address in the HOME statement to provide as the local address. The address that closely precedes a physical IP address will be used as the local address. For example:

```
HOME
  172.2.1.1     VIPA1  ; <-- Source for ETH1 and TR1
  151.2.3.1     ETH1
  151.4.1.1     TR1
  172.2.1.2     VIPA2  ; <-- Source for ETH2 and TR2
  151.2.3.2     ETH2
  151.4.1.2     TR2
```

Optionally, additional VIPA addresses can be defined to associate a group of interfaces and serve as local addresses. In the example above, VIPA1 is associated with ETH1 and TR1, and VIPA2 is associated with ETH2 and TR2.

If an outbound datagram is not to contain a SOURCEVIPA address for a particular interface (that is, use a physical IP address), then use the following example:

```
HOME
  151.4.1.1     TR1    ; <-- No SOURCEVIPA for outbound on TR1
  172.2.1.1     VIPA   ; <-- Source for ETH1 and TR2
  151.2.3.1     ETH1
  151.4.1.2     TR2
```

Figure 12 shows the various TCP/IP protocols when a SOURCEVIPA address is used as a local address. ″Y″ indicates that the SOURCEVIPA address will be used as the local address, and ″N″ indicates that the SOURCEVIPA address will not be used as the local address.

Destination	ICMP	TCP	RAW	UDP
Local Interface	N	N	N	N
Local network	Y	Y	Y*	Y*
Remote network	Y	Y	Y*	Y*

'*' = Except for routing applications

Figure 12. Source VIPA Usage Chart

Related Topics

- “DEVICE and LINK Statements” on page 101
- “PRIMARYINTERFACE Statement” on page 235
- “BSDROUTINGPARMS Statement” on page 140

INCLUDE Statement

This statement causes another data set that contains profile configuration statements to be included and processed at the point it occurs in the including file stream.

Syntax

Parameters

data_set_name
A fully qualified data set name that identifies a sequential file. The sequential file can be a sequential data set or a PDS with the member name. It cannot be an HFS file.

Modifying

Modification is not applicable to this statement.

Examples

The following example shows a profile that includes two other profiles:

```
; Include device/link/home/gateway/start for appropriate device
INCLUDE USER.INCLUDE(CTC)
;
; ARP age of 20 minutes
ARPAGE 20
;
; Default assorted parms
ASSORTEDPARMS
   IGNOREREDIRECT
   NOFWD
   NOUDPQUEUELIMIT
ENDASSORTEDPARMS
;
; Default IP Config parms
IPCONF ARPTO 1200 CLAWUSED STOPON NODATAGR IGNORER REASSEMBL 60 TTL 64
;
; Default Interval
KEEPALIVEOPTIONS
   INTERVAL 120
ENDKEEPALIVEOPTIONS
;
; Include sockets maximum connectors
INCLUDE USER.INCLUDE.MAXCON
;
; Default TCP Config parms
TCPCONF INT 120 UNRESTRICTL TCPSENDB 65535 TCPRCVB 65535
;
; Default UDP Config parms
UDPCONF UNRESTRICTL UDPCHK UDPSENDB 65535 UDPRCVB 65535 NOUDPQ
```

The following is the INCLUDE file, USER.INCLUDE(CTC), that defines the CTC device:

```
; CTC Device
DEVICE CTC1     CTC     D00  IOBUFFERSIZE 32K
LINK   CTCD00   CTC  0  CTC1 IFSPEED 4500000
;
```

```
HOME  9.67.113.105   CTCD00 ; MVSVIC04 - MVSVIC02
;
; Direct routes
GATEWAY
; Network  First hop  Driver  Packet size  Subnet mask    Subnet value
9.67.116.124   =      CTCD00       576        HOST
;
 START CTC1
```

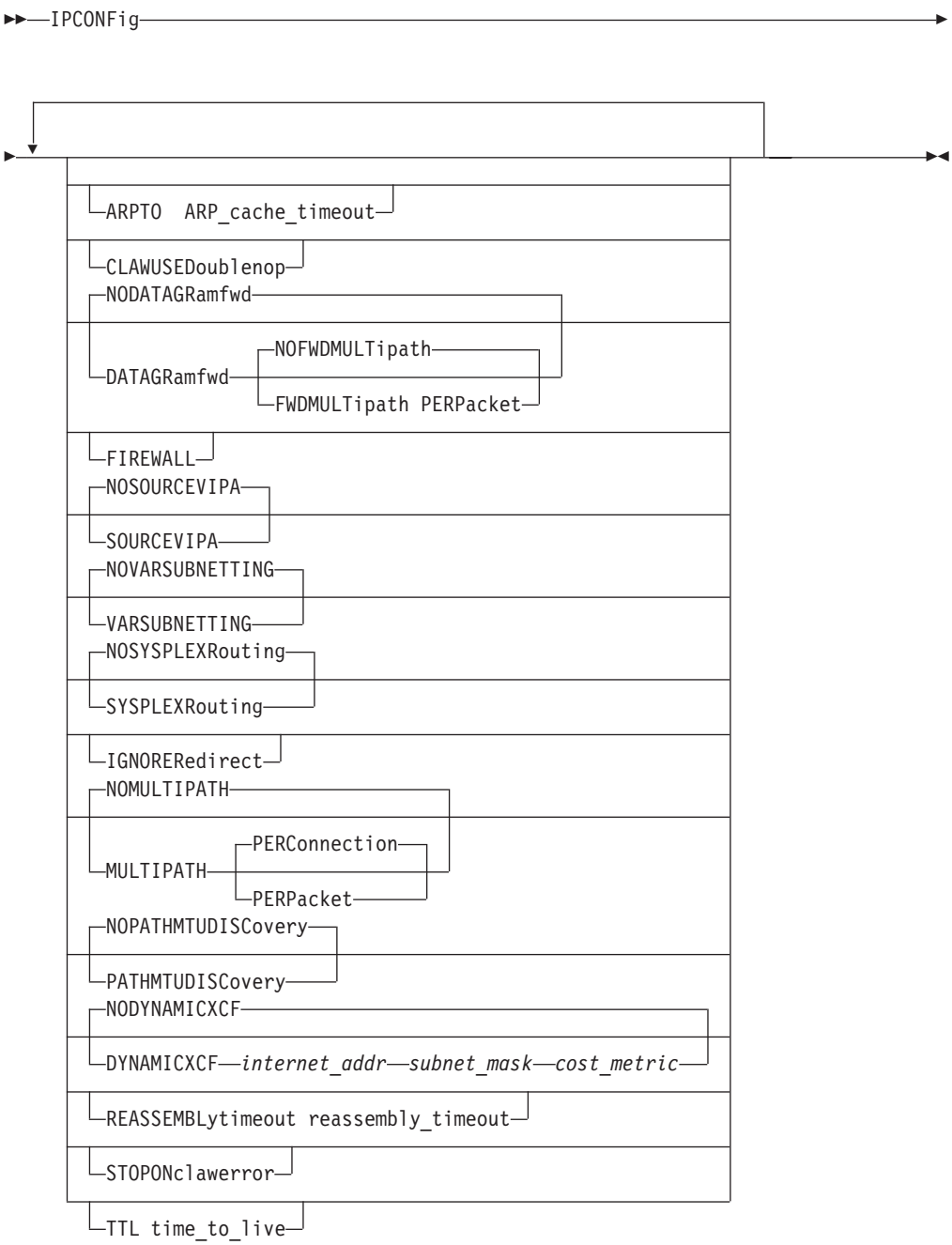
The following is the INCLUDE file, USER.INCLUDE.MAXCON, that defines maximum socket connections:

```
; Socket connections default is 10, want to run with more
SOMAXCON 50
```

IPCONFIG Statement

Use the IPCONFIG statement to update the IP layer of TCP/IP.

Syntax



Parameters

ARPTO

Use ARPTO to specify the number of seconds between creation or revalidation

and deletion of ARP table entries. The default is 1200 seconds. An ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address.

This parameter serves the same purpose as the ARPAGE statement, but the value specified on ARPAGE is in minutes while the value specified on the ARPTO parameter is in seconds.

Because ARP cache entries for MPCIPA and MPCOSA interfaces are not managed by the TCP/IP stack, they are not affected by the ARPTO statement.

CLAWUSEDOUBLENOP
Forces channel programs for CLAW devices to have two NOP CCWs to end the channel programs. This is required for some vendor devices, and applies to only first-level MVS systems. The CLAWUSEDOUBLENOP parameter is confirmed by the message:

```
CLAWUSEDOUBLENOP is set
```

NODATAGRAMFWD
Stops the transfer of data between networks by disabling IP datagram routing between different network interfaces. This statement can be used for security or to ensure correct usage of limited resources. The NODATAGRAMFWD parameter is confirmed by the message:

```
IP forwarding is disabled.
```

If either ASSORTEDPARMS NOFWD or IPCONFIG NODATAGRAMFWD is specified in a profile, or if neither the ASSORTEDPARMS nor the IPCONFIG statement is specified, forwarding is disabled. If the ASSORTEDPARMS or IPCONFIG statement is specified and the NOFWD and NODATAGRAMFWD parameters are not specified, forwarding is enabled.

DATAGRAMFWD
Enables the transfer of data between networks. The DATAGRAMFWD parameter is confirmed by the message:

```
IP forwarding is enabled
```

Notes:

1. The FWDMULTIPATH keyword on DATAGRAMFWD is independent of the MULTIPATH keyword on the IPCONFIG statement.
2. You can dynamically change the multipath settings DATAGRAMFWD using the OBEYFILE commands.

NOFWDMULTIPATH
In transferring data between networks, if there are multiple equal-cost paths to a destination and NOFWDMULTIPATH is specified, TCP/IP will use the first active route found for forwarding each IP packet. This is the default. The NOFWDMULTIPATH parameter is confirmed by the message:

```
IP Forwarding NoFwdMultipath support is enabled
```

FWDMULTIPATH PERPACKET
In transferring data between networks, if there are multiple equal-cost routes to a destination network or host, TCP/IP, upon forwarding an IP packet to a given host in that destination network, selects a route on a round-robin basis from a multipath routing list to that destination host. The selected route is used for routing that IP packet. Connection or connectionless oriented IP packets using the same destination address do not always use the same route, but they will use all possible active routes to that destination host. All IP packets for a given

association with a destination host are spread across the multiple equal-cost routes. The DATAGRAMFWD FWDMULTIPATH PERPACKET parameter is confirmed by the message:

```
IP Forwarding FwdMultipath PerPacket support is enabled
```

FIREWALL
Specifies that this TCP/IP host is to be used as a network firewall (filtering, tunnelling, or network address translation). The FIREWALL parameter is confirmed by the message:

```
Firewall support is enabled.
```

Firewall functions can only be activated at initial activation of TCP/IP.

NOSOURCEVIPA
Specifies that TCP/IP will not request to use the corresponding virtual IP address in the HOME list as the source IP address for outbound datagrams. The NOSOURCEVIPA parameter is confirmed by the message:

```
Source Vipa support is disabled.
```

NOSOURCEVIPA is the default.

SOURCEVIPA
Requests TCP/IP to use the corresponding virtual IP address in the HOME list as the source IP address for outbound datagrams. This parameter has no effect on RIP packets used by RIP services (OROUTED, NCPROUTE, or OMPROUTE) as well as OSPF packets by OSPF services (OMPROUTE). The SOURCEVIPA parameter is confirmed by the message:

```
Source Vipa support is enabled.
```

NOVARSUBNETTING
Specifies that variable subnetting is not being used. The NOVARSUBNETTING parameter is confirmed by the message:

```
Variable subnetting is disabled.
```

NOVARSUBNETTING is the default.

VARSUBNETTING
Specifies that variable subnetting is being used. Enables variable subnetting and supernetting support in TCP/IP. Variable-length subnet masks may be coded on the GATEWAY and BSDROUTINGPARMS statements. Also, this option allows OMPROUTE and OROUTED applications to dynamically update the IP routing table with variable-length subnet masks. If OROUTED is configured to use RIPv2, VARSUBNETTING must be enabled. If OMPROUTE is started, VARSUBNETTING will be enabled automatically. The VARSUBNETTING parameter is confirmed by the message:

```
Variable subnetting is enabled.
```

NOSYSPLEXRouting
Specifies that this TCP/IP host is not part of an MVS sysplex domain. The NOSYSPLEXROUTING parameter is confirmed by the message:

```
SysplexRouting support is disabled.
```

NOSYSPLEXROUTING is the default.

SYSPLEXRouting
Specifies that this TCP/IP host is part of an MVS sysplex domain and should communicate interface changes to the workload manager (WLM). The SYSPLEXROUTING parameter is confirmed by the message:

```
SysplexRouting support is enabled.
```

IGNOREREDIRECT
Causes TCP/IP to ignore ICMP Redirect packets. The IGNOREREDIRECT parameter is confirmed by the message:

```
ICMP will ignore redirects
```

If you are using OROUTED, use this option because OROUTED does not support ICMP redirects.

NOMULTIPATH
Disables the multipath routing selection algorithm for outbound IP traffic. If there are multiple equal-cost routs to a destination and NOMULTIPATH is specified, TCP/IP will use the first active route found for sending each IP packet. This is the default. The NOMULTIPATH parameter is confirmed by the message:

```
Multipath support is disabled
```

NOMULTIPATH is the default.

MULTIPATH
Enables the multipath routing selection algorithm for outbound IP traffic. In general, multipath routing provides the routing distribution necessary to balance the network utilization of outbound packets by load splitting. Multipath routing requires the definition of multiple equal-cost routes which are either defined statically or added dynamically by routing protocols. If MULTIPATH is specified without any subparameters, the default is PERCONNECTION. The MULTIPATH parameter has no effect if there are no multipath routes in the TCP/IP configuration.

Note: In some cases, it might appear that data is not being equally distributed among each of the equal-cost interfaces. This depends upon the characteristics of the application which is sending or receiving data. For example, when osnmp walk is issued, the application initially sends data using a source IP address of INADDR_ANY. Subsequently, when the application receives a response, all future sends use the source IP address of the interface where data was just received. The result is all data that is sent out is on a single interface, which is independent of any multipath setting.

PERCONNECTION
If there are multiple equal-cost routes to a destination network or host and MULTIPATH PERCONNECTION is specified, TCP/IP, upon first sending an IP packet to a given host in that destination network or host, will select a route on a round-robin basis from a multipath routing list to that destination host. The selected route will be used for routing IP packets for a given connection or connectionless oriented association to that destination host. Connection or connectionless oriented IP packets using the same association will always use the same route as long as that route is active. All associations with a given host will be spread across the multiple equal-cost routes. The MULTIPATH PERCONNECTION parameter is confirmed by the message:

```
Multipath PerConnection support is enabled
```

PERPACKET
If there are multiple equal-cost routes to a destination network or host, TCP/IP, upon sending an IP packet to a given host in that destination network, will select a route on a round-robin basis from a multipath routing list to that destination host. The selected route will be used for routing that IP packet. Connection or connectionless oriented IP packets using the same source and

destination address pair will not always use the same route, but they will use all possible active routes to that destination host. All IP packets for a given association with a destination host will be spread across the multiple equal-cost routes. The MULTIPATH PERPACKET parameter is confirmed by the message:

```
Multipath PerPacket support is enabled
```

Notes:

1. Use this option only as an attempt to improve aggregate throughput of IP traffic over multipath routes and that potentially high CPU consumption in reassembly of out-of-order packets at the receiving end is not an issue. Performance varies according to network configurations used.
2. Care in using this option must be made with Firewall configurations. Packet loss might occur over non-secured or secured interfaces that are used by the multipath routes.
3. IP traffic on RSVP-based routes are restricted from using this option. Instead, the PerConnection option is used for these routes.

NOPATHMTUDISCOVERY

Indicates that TCP/IP is not to provide path MTU (PMTU) discovery support. This is the default. The NOPATHMTUDISCOVERY parameter is confirmed by the message:

```
PATHMTUDISCOVERY support is disabled
```

NOPATHMTUDISCOVERY is the default.

PATHMTUDISCOVERY

Indicates that TCP/IP is to dynamically discover the path MTU, which is the minimum of the MTUs of each hop in the path. Use this parameter to prevent fragmentation of datagrams. The PATHMTUDISCOVERY parameter is confirmed by the message:

```
PATHMTUDISCOVERY support is enabled
```

NODYNAMICXCF

Indicates that XCF dynamic support is not enabled. The NODYNAMICXCF parameter is confirmed by the message:

```
DYNAMIC XCF DEFINITIONS ARE DISABLED
```

NODYNAMICXCF is the default.

DYNAMICXCF *internet_addr subnet_mask cost_metric*

Indicates that XCF dynamic support is enabled. The *internet_addr* is the IP address to be used for HOME statements for all dynamically-generated XCF or same-host links. The *subnet_mask* and *cost_metric* will be used on a BSDROUTINGPARMS entry for OROUTED. If dynamic routing is being provided by OMPROUTE, the *subnet_mask* and *cost_metric* values must be configured to OMPROUTE using the Interface, OSPF_Interface, or RIP_Interface configuration statements in the OMPROUTE configuration file. The DYNAMICXCF parameter is confirmed by the message:

```
DYNAMIC XCF DEFINITIONS ARE ENABLED
```

Note: The VTAM ISTLSXCF major node must be active for XCF dynamics to work, except for TCPs on the same MVS image; a dynamic samehost definition will be generated regardless of whether ISTLSXCF is active or not. For information on activating the ISTLSXCF major node, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

REASSEMBLYTIMEOUT *reassembly_timeout*
IP reassembly time-out value in seconds.

STOPONCLAWERROR
Stops channel programs (HALTIO and HALTSIO) when a device error is detected. The STOPONCLAWERROR parameter is confirmed by the message:

```
STOPONCLAWERROR is set
```

TTL *time_to_live*
IP time to live or hop count.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

OS/390 Firewall functions may not be activated via VARY OBEYFILE on an active TCP/IP stack. To activate OS/390 Firewall Technologies, halt all traffic on the designated TCP/IP stack, stop the stack, modify the TCP profile to include 'IPCONFIG FIREWALL', and restart the stack.

Examples

This example shows an IPCONFIG statement that causes ARP table entries to be deleted 2400 seconds after creation or revalidation, forces channel programs for CTC devices to have two NOP CCWs to end the channel programs, disables IP forwarding, and causes TCP/IP to halt on certain CLAW errors.

```
IPCONFIG ARPTO 2400 CLAWUSED NODATAGR STOPON
```

Usage Notes

- If any of the IPCONFIG statement parameters are specified on other statements, such as ASSORTEDPARMS, the settings from the last statement processed are used. You will see an appropriate message.
- When DYNAMICXCF is coded in the profile, the purpose is to generate those dynamic XCF devices/links if it is possible. When TCP/IP is up, but ISTLSXCF is not active, dynamic creation is deferred until later when there is a TCP/IP command (for example, V OBEY, V start/stop command) to start the internal profile processing, and the dynamic devices/links will be generated at that time.
- NOVARSUBNETTING cannot be specified after a BSDROUTINGPARMS or GATEWAY statement is specified with supernetwork subnet masks.
- Firewall normally requires additional configuration. Refer to *OS/390 Firewall Technologies, Guide and Reference* (SC24-5835) for additional information.
- If dynamic XCF definitions have been enabled but a profile for VARY TCPIP,,OBEYFILE contains NODYNAMICXCF, only future dynamic definitions are affected. Existing definitions and connectivity are not affected.
- Dynamic XCF definitions will not be generated when a device name exists that corresponds to a newly-discovered VTAM. Also, dynamic SameHost defintions will not be generated when a DEVICE IUTSAMEH already exists.
- Currently, you can also disable IP forwarding with the ASSORTEDPARMS statement. If your profile contains the ASSORTEDPARMS statement, configuration proceeds as if DATAGRAMFWD NOFWDMULTIPATH is specified on IPCONFIG.
- If PERPACKET is disallowed for multipath routes (for example, Firewall security), the IP-forwarded datagrams are not subjected to multipath.

- If you do not include any configuration data in the OMPROUTE configuration file for the XCF links, OMPROUTE will not communicate a routing protocol (OSPF or RIP) over the interfaces. OMPROUTE will include (in the data sent to other routers) information relative to the XCF link(s) as long as Send_Static_Routes=YES is configured for RIP Interfaces and AS_Boundary_Routing(Import_Static_Routes=YES) is configured for OSPF.
- If you want to communicate the OSPF or RIP protocol over a subset of the XCF links, you must configure the appropriate links in the OMPROUTE configuration file using the OSPF_Interface or RIP_Interface statements. Doing this will allow OMPROUTE to communicate to other routers not only the information relative to the XCF link(s), but also information relative to resources on the other side of the host at the opposite end of the XCF link(s).

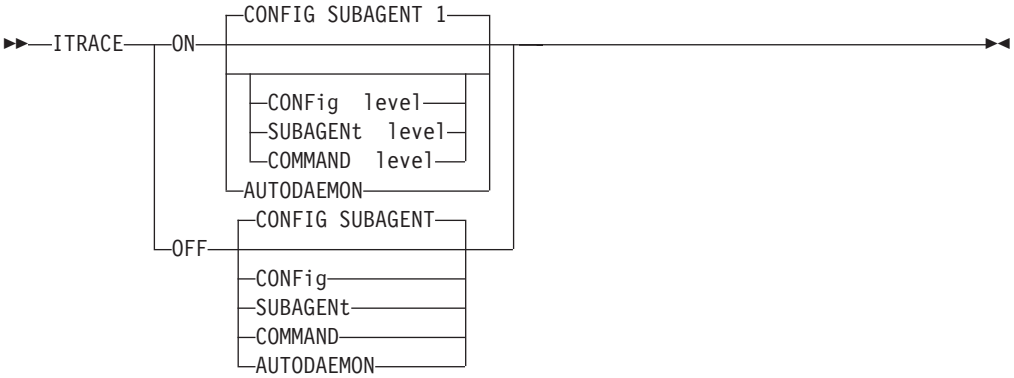
  To configure the appropriate links, you can explicitly configure each XCF link as either an OSPF or RIP interface (including those that might become active in the future). Alternatively, you can use the wildcard configuration capability of OMPROUTE to configure your XCF links.

  To use the wildcard configuration, use a wildcard address (for example, 9.67.100.*) on the OSPF_Interface or RIP_Interface statement instead of an explicit address. In this way, any interface address that falls within that wildcard (9.67.100.1, 9.67.100.2, and so on) will be configured using the parameters specified on the wildcard definition statement.

ITRACE Statement

Use the ITRACE statement to control TCP/IP run-time tracing.

Syntax



Parameters

ON
Select ON to establish run-time tracing.

OFF
Select OFF to terminate run-time tracing.

CONFIG
Turn internal trace for configuration ON or OFF.

SUBAGENT
Turn internal trace for SNMP subagent ON or OFF.

COMMAND
Turn internal trace for command ON or OFF.

AUTODAEMON
Turn internal trace for the autolog subtask ON or OFF.

level
Indicates the tracing level to be established. Levels are as follows:

Levels for CONFIG

1 ITRACE for all of config

2 General level of tracing for all of config

3 Tracing for configuration set commands

4 Tracing for configuration get commands

5 Tracing for syslog calls issued by config

100 Tracing for the parser

200 Tracing for scanner

300 Tracing for mainloop

400 Tracing for commands

Levels for SUBAGENT

1 General subagent tracing

2 General subagent tracing plus DPI traces

3 General subagent tracing plus extended storage dump traces

4 All trace levels

Levels for COMMAND

1 ITRACE for all commands

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

```
ITRACE ON CONFIG 3
ITRACE OFF SUBAGENT
```

Usage Notes

- This statement is used primarily for diagnostic purposes.
- Subagent trace output is directed to the syslog daemon. This daemon is configured by the */etc/syslog.conf* HFS file and must be active.
- AUTOLOG trace output goes to ALGPRINT.
- CONFIG trace output goes to CFGPRINT. If CFGPRINT is not coded in TCPIPROC, the Config component dynamically allocates a DDName CFGPRINT.
- Command trace output goes to the hardcopy console log.
- ITRACE ON commands are cumulative until an ITRACE OFF is issued.
- ITRACE should only be set at the direction of an IBM Service representative.

Related Topics

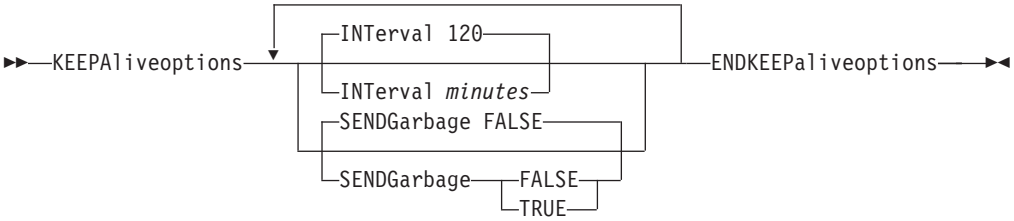
Refer to *OS/390 SecureWay Communications Server: IP Diagnosis* for more information.

KEEPALIVEOPTIONS Statement

Use the KEEPALIVEOPTIONS statement to specify the operating parameters of the TCP keep-alive packets. The parameters apply to all TCP connections for which keep-alive has been activated through the setsockopt() call of the C socket interface.

Note: Support for the KEEPALIVEOPTIONS statement will be dropped in a future release. It is recommended that you use the INTERVAL and SENDGARBAGE parameters on the TCPCONFIG statement instead of KEEPALIVEOPTIONS.

Syntax



Parameters

INTERVAL *minutes*
The number of minutes TCP waits after receiving a packet for a connection before it sends a keep-alive packet for that connection. The range is from 0 to 35791 minutes. A value of 0 will disable the keepalive function.

SENDGARBAGE
Specifies whether the keep-alive packets sent by TCP contain 1 byte of random data.

FALSE
Causes the packet to contain no data. This is the default.

TRUE
Causes the packet to contain 1 byte of random data and an incorrect sequence number, assuring that the data is not accepted by the remote TCP.

Modifying

To change a parameter value, you must respecify the statement with the new parameter value. Any parameters not specified will be reset to their default value.

Usage Notes

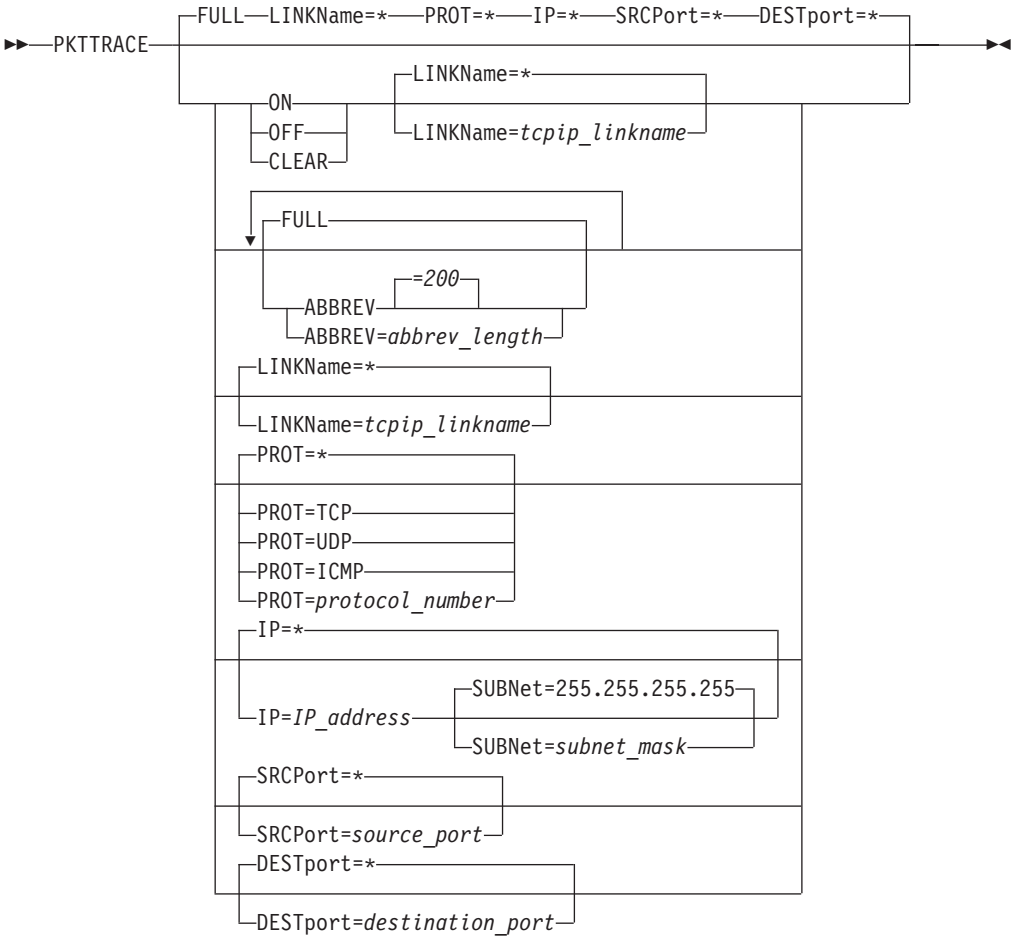
- KEEPALIVE INTERVAL 0 turns off TCP keepalive.
- KEEPALIVE INTERVAL is a global setting for the entire TCP stack. Each socket can select keepalive with setsocketopt().
- The ENDKEEPALIVEOPTIONS statement specifies the end of the KEEPALIVEOPTIONS information. If it is omitted, subsequent entries will generate error messages.

PKTTRACE Statement

Use the PKTTRACE statement to control the packet tracing facility in TCP/IP. You can use this statement to select IP packets as candidates for tracing and subsequent analysis. An IP packet must meet all the conditions specified on the statement for it to be traced.

The PKTTRACE statement consists of two parts. The first part defines to TCP/IP the links that are to be traced and characteristics of how they are to be traced. The second part turns packet tracing ON or OFF, or CLEARs packet trace settings for the links specified on prior PKTTRACE statements or for a single link if the LINKName parameter is used.

Syntax



Parameters

ABBREV

Specifies that a truncated portion of the IP packet is to be traced. You can specify a length between one and 65535 or use the default of 200. The ABBREV parameter can be used to reduce the volume of data stored in the trace file.

CLEAR
Disables packet tracing for the links specified and removes the characteristics defining how they should be traced.

DESTPORT
Specifies a port number that will be compared with the destination port of inbound and outbound packets. The port number is an integer between one and 65535. If the destination port of a packet is the same as the specified port number, the packet will be traced. This comparison is only performed for packets using either the TCP or UDP protocol; packets using other protocols are not traced. If the DESTPORT parameter is omitted, there is no checking of the destination port of packets. If an asterisk (*) is specified, packets of any protocol and any source port will be traced.

FULL
Specifies that the entire IP packet is to be traced.

IP
Specifies an IP address that will be compared with both the source and destination addresses of inbound and outbound packets. If either the source or destination address of a packet matches the specified IP address, the packet will be traced. The IP address must be specified in dotted decimal notation. If the IP option is omitted, or an asterisk (*) is specified, then all IP addresses will be traced.

LINKNAME
Specifies the name of the link defined in the preceding LINK statement. If the LINKNAME parameter is omitted or an asterisk (*) is specified, the PKTTRACE parameters will apply to all links prior to this statement.

To facilitate defining packet tracing when many links are involved, use the PKTTRACE statement with the LINKNAME=* option to define packet tracing characteristics for the majority of the links. Then use individual PKTTRACE statements with specific LINKNAME parameters for each link that must be defined differently from the majority.

OFF
Disables packet tracing for the links specified and removes the characteristics defining how they should be traced.

If LINKNAME=* and all other parameters are defaults, all trace structures are deactivated and removed from all existing links.

If LINKNAME=* and PROT=UDP, all trace structures for all resources are analyzed; any matches are removed. If no trace structures remain, trace is deactivated for that resource.

If LINKNAME=*link_name* and there are no other parameters, all trace structures for *link_name* are deactivated and removed.

If LINKNAME=*link_name*,IPADR=127.0.0.1, that particular trace structure is removed if it is found. If there is only one trace structure, then that structure is removed and trace is deactivated for that resource.

ON
Turns on packet tracing, clears all settings previously defined and refreshes just the default settings.

If you use LINKNAME=* and all other parameters are defaults, even if the defaults are specified, the command results replaces any existing trace structures for all existing links.

If you use LINKNAME=*link_name* and another non-default parameter, the command results are added to any existing trace structures. However, if the existing trace structure for *link_name* is all defaults, the existing trace structure will be discarded.

PROT
Specifies the protocol type to be traced. This can be specified as one of the literals TCP, UDP, or ICMP, or as a number between zero and 255 (ICMP=1, TCP=6, UDP=17, and RAW=255). If the PROT parameter is omitted or an asterisk (*) is specified, packets of any protocol will be traced.

SRCPORT
Specifies a port number that will be compared with the source port of inbound and outbound packets. The port number is an integer between one and 65535. If the source port of a packet is the same as the specified port number, the packet will be traced. This comparison is only performed for packets using either the TCP or UDP protocol; packets using other protocols are not traced. If the SRCPORT parameter is omitted, there is no checking of the source port of packets. If an asterisk (*) is specified, packets of any protocol and any source port will be traced.

SUBNET
Specifies a subnet mask that applies to the host and network portions of the IP address specified on the accompanying IP parameter. The subnet mask must be specified in dotted decimal notation and must be specified in conjunction with the IP parameter.

Modifying

You can enter PKTTRACE commands into an OBEYFILE at any time. However, the commands must be entered after the links have been defined. For example:

```
PKTTRACE ON,LINKNAME=*
LINK ...
DEVICE ...
```

In the above example, the trace is only done for the LOOPBACK interface.

For more information about changing PKTTRACE parameters, see the descriptions for the ON and OFF parameters for PKTTRACE above.

Multicast routes can be specified using host specification. A general multicast default route can be specified using the multicast group address of 224.0.0.0:

```
GATEWAY

;Host      First hop  Link   packet size

 224.0.0.0  =         LINK1  DEFAULTSIZE  HOST
```

Specific multicast group routes may also be specified:

```
GATEWAY

;Host      First hop  Link   packet size

 224.1.1.1  =         LINK2  DEFAULTSIZE  HOST
```

The order of precedence for determining the route of an outbound multicast datagram is as follows:

1. Route by interface using MSG_DONTROUTE with the send(), sendmsg(), or sendto() calls.
2. Application uses setsockopt() IP_MULTICAST_IF to specify the interface to use.
3. Specific multicast group route specified.
4. General multicast default group address specified (224.0.0.0).
5. If DEFAULTNET is specified and the interface is multicast capable, this link will be used.

Given the preceding two sample GATEWAY statements and assuming the application does not use route by interface and does not code the setsockopt() IP_MULTICAST_IF:

- If an application sends a datagram to 224.2.2.2, LINK1 will be used.
- If an application sends a datagram to 224.1.1.1, LINK2 will be used.

More than one PKTTRACE statement can be included in the TCP/IP profile or using the VARY TCPIP command. Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (“*”), which indicates all links. When multiple statements are included, the last statement processed will be in effect. If each statement has a specific link name, then the changes only affect those specific links. Otherwise, all links are affected. Parameters not specified will be defaulted. For example, if you specified:

```
PKTTRACE ON
PKTTRACE PROT=UDP
PKTTRACE IP=127.0.0.1
```

Packet trace would be active for all link names, all protocols, all SUBNETs, all SRCPORTs, and all DESTPORTs by default, and for IP address 127.0.0.1. The first PROT=UDP is reset by the default PROT=* on the second PKTTRACE statement. To get both you would have to specify:

```
PKTTRACE ON
PKTTRACE PROT=UDP IP=127.0.0.1
```

PKTTRACE statements that affect all linknames will affect links that have trace turned OFF but have not been cleared.

Usage Notes

- IP=* implies IP=0.0.0.0 and SUBNET=255.255.255.255.
- The IP address and SUBNET pair specified must be in the same network.
- If a given keyword is specified multiple times, the last value specified is used. If an option appears more than once on a statement, the value associated with the last occurrence of the option is used.
- Packet traces are recorded externally using the IPCS CTRACE writer instead of GTF. See *OS/390 SecureWay Communications Server: IP Diagnosis* for information on the steps required to perform an IP packet trace.
- The PKTTRACE statement must appear after a valid LINK statement for the link in the PROFILE.TCPIP data set.
- CS for OS/390 allows a single TCP/IP address space to drive many devices, including more than one instance of any particular device, to provide connections to the TCP/IP network. The PKTTRACE statement supports this capability through the LINKNAME option.

- Options on the statement can appear in any order.
- If no options are specified on the PKTTRACE command, then all packets through all devices will be traced.
- If an error is found while parsing the PKTTRACE statement, an error message is generated, the parameter in error is ignored, and the rest of the statement is parsed. If an error is produced by an incorrect ABBREV value, the ABBREV value is changed to the default.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created, or reset using the CLEAR option, a link's trace profile is set to the default values for the trace options as follows:

  PROT
  All protocols

  IP All IP addresses

  SUBNET
  No checking

  SRCPORT
  No checking

  DESTPORT
  No checking

  FULL
  Tracing of the whole IP packet

Examples

The following sample includes several examples of the PKTTRACE statement:

```
; CTC Device and Link
DEVICE CTC1     CTC     D00
LINK   CTCD00   CTC  1  CTC1
;
; CTC Device and Link
DEVICE CTC2     CTC     D02
LINK   CTCD02   CTC  1  CTC2
;
; CTC Device and Link
DEVICE CTC3     CTC     D04
LINK   CTCD04   CTC  1  CTC3
;
; LCS Device and Links
DEVICE LCS1     LCS     100
LINK   TR1      IBMTR        1  LCS1
LINK   LCSC00   ETHERNET     2  LCS1
LINK   LCSF00   FDDI         3  LCS1
;
DEVICE LCS2     LCS     102
LINK   LCS802   802.3        1  LCS2
;
DEVICE LCS3     LCS     104
LINK   LCSE802  ETHEROR802.3 1  LCS3
;
; start pkttrace
PKTTRACE ON LINKNAME=*
;
; set defaults for all links not specified below
PKTTRACE
; set for CTCD00
PKTTRACE FULL LINKNAME=CTCD00 PROT=* IP=* SRCPORT=* DESTPORT=*
```

```
; set for CTCD02
PKTTRACE ABBREV LINKNAME=CTCD02 PROT=TCP IP=9.67.116.124
              SRCPORT=5000 DESTPORT=161
; set for CTCD04
PKTTRACE ABBREV=1 LINKNAME=CTCD04 PROT=UDP IP=9.67.116.124
              SUBNET=255.255.255.255 SRCPORT=161 DESTPORT=5000
; set for TR1
PKTTRACE ABBREV=200 LINKNAME=TR1 PROT=ICMP IP=*
              SRCPORT=5000 DESTPORT=161
; set for LCSC00
PKTTRACE ABBREV=65535 LINKNAME=LCSC00 PROT=1 IP=9.67.116.124
              SUBNET=255.255.255.255 SRCPORT=* DESTPORT=*
; set for LCSF00 not to trace
PKTTRACE OFF LINKNAME=LCSF00
```

Related Topics

Refer to *OS/390 SecureWay Communications Server: IP Diagnosis*.

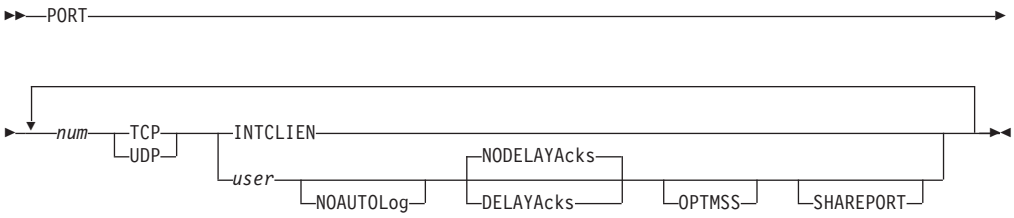
PORT Statement

Use the PORT statement to reserve a port for specified user IDs, procedures, or job names. The PORT statement also specifies the protocol to be used, if the *user* should not be autologged, and if TCP protocol acknowledgements should be delayed.

Note: By default the stack looks for the job name when comparing with the port and portrange reservation, but under TSO, the job name is set to the user ID. In those cases, putting a user ID on the reservation does work if the bind() call is made from TSO. Under OMVS, the job name is sometimes the user ID. For a started procedure, the job name is set to the procedure name. Also, you can use the name of the started JCL procedure for the UNIX System Services Kernel Address Space to allow almost any caller of the bind() socket API (except for users of the Pascal API) to bind to the port. This name is typically OMVS unless a different name is explicitly specified in the STARTUP_PROC parameter in the BPXPRMxx parmlib member. Refer to the *OS/390 MVS Initialization and Tuning Reference* for more details on the STARTUP_PROC parameter.

The port options (for example, NOAUTOLOG, DELAYACKS, OPTMSS, and so on) must be specified in the same order as they appear on the syntax diagram shown below.

Syntax



Parameters

num
Reserves a port for one or more users. The same port number can appear in more than one PORT statement with different users or more than once in the same PORT statement. This port cannot appear in a range specified by the PORTRANGE statement. If a PORTRANGE statement that includes this port number is specified prior to this statement, this port is ignored. If the PORTRANGE statement follows this statement, then the PORTRANGE statement is ignored. An error message is generated in either case. *num* is a value in the range 1–65535.

INTCLIEN
If the PORT statement has a value of INTCLIEN, it assigns the port to the internal Telnet server rather than assigning it to a client. Therefore, you must use the same port number as specified on the INTERNALCLIENTPARMS statement. INTCLIEN is only valid with the TCP protocol. OPTMSS and SHAREPORT are not valid with INTCLIEN.

user
For TCP, this specifies one or more job names that can use the given port. This allows multiple users to do a passive open on a well-known port and allows

multiple servers. When using a procedure name, it must be the member name of the cataloged procedure you use to start the address space, not the name on the EXEC statement in the procedure.

For UDP, only one job name can be associated with a given port.

NOAUTOLOG
Tells the TCP/IP address space ***not*** to restart the server if it was stopped previously. Otherwise, the default is to restart the server if it was stopped previously.

DELAYACKS
Allows you to alter the default TCP/IP behavior for acknowledgements and delay their transmission so that they can be combined with data sent to the foreign host. This affects acknowledgements returned when a packet is received with the PUSH bit on in the TCP header. The default behavior is to return an acknowledgement immediately.

The DELAYACKS parameter on the PORT or PORTRANGE statement only applies to the TCP protocol and only affects acknowledgements on this port connection.

NODELAYACKS
Specifies that an acknowledgement is returned immediately. This is the default.

OPTMSS
Specifies that the optimal maximum segment size function is on. OPTMSS is only valid for TCP ports.

SHAREPORT
Required when reserving a port to be shared across multiple listeners. If multiple jobnames are reserved for the same port, SHAREPORT only needs to be specified on one instance of the port reservation. SHAREPORT is only valid for TCP ports.

When SHAREPORT is specified, TCP/IP will allow multiple listeners to listen on the same port. As incoming client connections arrive for this port, TCP/IP will distribute them across the listeners. TCP/IP will select the listener with the least number of connections (both active and in the backlog) at the time that the incoming client connection request is received.

Modifying

To change a parameter value, you must delete the existing PORT statement, then redefine with the new PORT statement.

Examples

```
PORT
    20 TCP OMVS NOAUTOLOG              ; FTP server data port
    21 TCP FTPD1                       ; FTP server control port
    23 TCP INTCLIEN                    ; TELNET server
   111 TCP PORTMAP                     ; Portmap server
   111 UDP PORTMAP                     ; Portmap server
   161 UDP OSNMPD                      ; SNMP agent
   162 UDP SNMPQE                      ; SNMP Query Engine
   520 UDP OROUTED                     ; RouteD server
   620 TCP FTPD1 NOAUTOLOG OPTMSS
  1023 TCP PASCALS NOAUTOLOG SHAREPORT ; Pascal API server
  2000 TCP DFH41R5 NOAUTOLOG SHAREPORT ; CICS server 1 share port
```

Usage Notes

- A port that is not reserved in this list or with the PORTRANGE statement can be used by any user. If you have TCP/IP hosts in your network that use ports in the range 1–1023 for privileged applications, you should reserve them with this statement, the PORTRANGE statement, or the RESTRICTLOWPORTS parameter on the ASSORTEDPARMS, TCPCONFIG, or UDPCONFIG statements.
- For OS/390 UNIX applications, you can reserve a port by specifying the jobname of the application or the special jobname of OMVS.
- For syslogd, you must include the following PORT statement:

  ```
  PORT
      514 UDP OMVS         ; syslogd Server
  ```

  This port is required for syslogd to accept log data from remote syslogd servers.
- If you want SNMP ATM Management support, see “Step 4: Configure the ATM Open Systems Adapter 2 (ATM OSA-2) Support” on page 867 for an explanation of specifying the PORT statement.

Related Topics

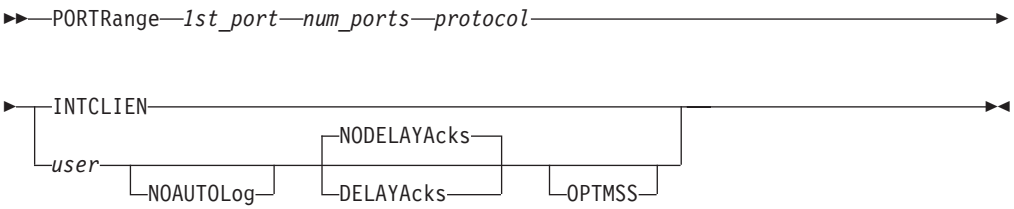
- “Specifying the TELNETPARMS Statements in the PROFILE Data set” on page 404
- “DELETE Statement” on page 144
- “GATEWAY Statement” on page 193
- “PORTRANGE Statement” on page 232
- “VARY Command—TCPIP Address Space” on page 264

PORTRANGE Statement

Use the PORTRANGE statement to reserve a range of ports for specified user IDs, procedures, or job names. The PORTRANGE statement also specifies the protocol to be used, if the *user* should not be autologged, and if TCP protocol acknowledgements should be delayed or not.

The portrange options (NOAUTOLOG, DELAYACKS, and OPTMSS) must be specified in the same order as they appear on the syntax diagram shown below.

Syntax



Parameters

1st_port
The starting port for a range of ports to reserve. The same port number can not appear in multiple PORTRANGE statements, nor can the port be specified on both PORTRANGE and PORT statements. If the port is specified on a PORT statement prior to this statement, this port range is ignored. If the port is specified on a PORT statement that follows this statement, the port in the PORT statement is ignored. An error message is generated in either case.*1st_port* is a value between 1 and 65535.

num_ports
The number of ports to reserve. The ports reserved cannot overlap other ranges specified by a PORTRANGE statement. No ports within this range can be specified on a PORT statement. If the port is specified on a PORT statement prior to this statement, this port range is ignored. If the port is specified on a PORT statement that follows this statement, the port in the PORT statement is ignored. An error message is generated in either case.

protocol
Specifies the protocol to be used, either TCP or UDP.

INTCLIEN
If the PORTRANGE statement has a value of INTCLIEN, it assigns the ports to the internal Telnet server rather than to a client. Therefore, you must use the same port numbers as specified on the INTERNALCLIENTPARMS statement. INTCLIEN is only valid with the TCP protocol. OPTMSS is not valid with INTCLIEN.

user
For TCP, this specifies one or more jobnames that can usethe given port. This allows multiple users to do a passive open on a well-known port and allows multiple servers. When using a procedure name, it must be the member name of the cataloged procedure you use to start the address space, not the name on the EXEC statement in the procedure.

For UDP, only one jobname can be associated with a given port.

NOAUTOLOG
Tells the TCP/IP address space ***not*** to restart the server if it was stopped previously. Otherwise, the default is to restart the server if it was stopped previously.

DELAYACKS
Allows you to alter the default TCP/IP behavior for acknowledgements and delay their transmission so that they can be combined with data sent to the foreign host. This affects acknowledgements returned when a packet is received with the PUSH bit on in the TCP header. The default behavior is to return an acknowledgement immediately.

The DELAYACKS parameter on the PORT or PORTRANGE statement only applies to the TCP protocol and only affects acknowledgements on this port connection.

NODELAYACKS
Specifies that an acknowledgement is returned immediately. This is the default.

OPTMSS
Specifies that the optimal maximum segment size function is on. OPTMSS is only valid for TCP ports.

Modifying

To change a parameter value, you must delete the existing PORTRANGE statement, then redefine with the new PORTRANGE statement.

Examples

This example shows a PORTRANGE statement used to reserve a large number of ports for a single test system.

```
PORTRANGE
    4000 200  TCP TESTSYS
```

This example shows a PORTRANGE statement where port 111 is reserved for both UDP and TCP for one user, and ports 500-504 are reserved for two different users, one using UDP and one using TCP. Note that for multiple users to share the same TCP port, a PORT statement is required. Multiple users cannot share the same UDP port.

```
PORTRANGE
     111   1  UDP  PORTMAP
     111   1  TCP  PORTMAP
     500   5  UDP  USER1
     500   5  TCP  USER2

PORT 600 TCP USER1
     601 TCP USER1
     602 TCP USER1
     600 TCP USER2
     601 TCP USER2
     602 TCP USER2
     600 TCP USER3
     601 TCP USER3
     602 TCP USER3
```

Usage Notes

- A port that is not reserved by a PORT or PORTRANGE statement can be used by any user. If you have TCP/IP hosts in your network that reserve ports in the range 1-1023 for privileged applications, you should reserve them either with this

statement, the PORT statement, or the RESTRICTLOWPORTS parameter on the ASSORTEDPARMS, TCPCONFIG, or UDPCONFIG statements.

- If you are reserving ports for the INADDRANYPORT() parameter in the BPXPRMxx parmlib member, you need to specify a jobname of OMVS for that PORTRANGE statement.

Related Topics

- "DELETE Statement" on page 144
- "PORT Statement" on page 229
- "VARY Command—TCPIP Address Space" on page 264

PRIMARYINTERFACE Statement

Use the PRIMARYINTERFACE statement to specify which link is to be designated as default local host for use by the GETHOSTID() function.

Except for the SOURCEVIPA option in the ASSORTEDPARMS or IPCONFIG statements, the PRIMARYINTERFACE has no effect on outbound traffic to a network when there are multiple network interfaces. For more information on how outbound traffic is routed to a network in the presence of multiple network interfaces, see “Multiple Network Attachments Support” on page 104.

Syntax

Parameters

link_name
The name of a link as defined in a LINK statement that is to be the primary interface. This link must have been also defined in a previous HOME statement.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example shows a PRIMARYINTERFACE statement specifying a token ring:

```
PRIMARYINTERFACE TR1
```

You can verify which HOME entry is primary by using the onetstat -h command:

```
Home address list:
Address          Link             Flg
9.67.113.61      TR1              P
9.67.116.125     CTCD00
127.0.0.1        LOOPBACK
```

Usage Notes

- The primary interface is flagged in the onetstat -h display.
- The primary interface IP address is used as the source IP 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 will be designated as the default local host.

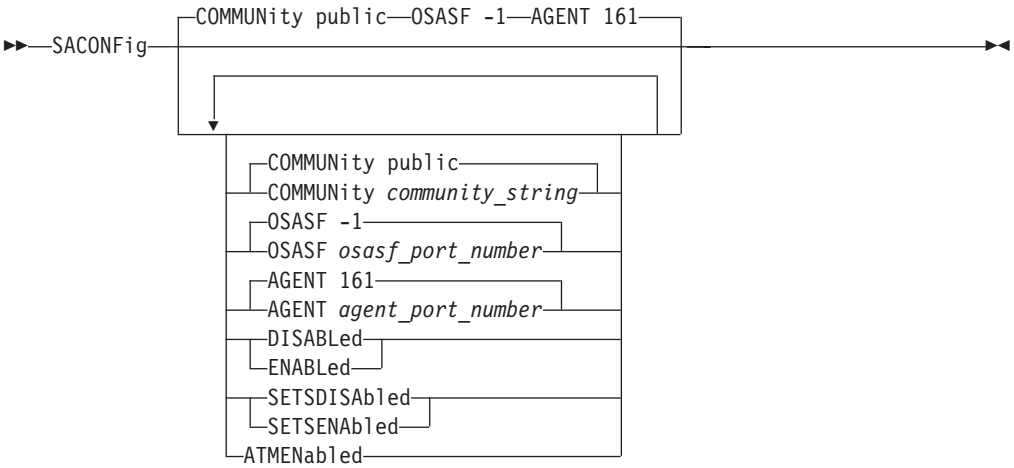
Related Topics

- “DEVICE and LINK Statements” on page 101
- “HOME Statement” on page 207

SACONFIG Statement

Use the SACONFIG statement to configure the SNMP subagent.

Syntax



Parameters

COMMUNITY
A character string of up to 32 characters used as the community name (or password) in establishing contact with the SNMP agent. For the TCP/IP SNMP subagent to communicate with the CS for OS/390 SNMP agent, the community name specified (or defaulted) on the COMMUNITY keyword must match one that is defined in the PW.SRC or SNMPD.CONF data set used by the SNMP agent or specified (or defaulted) on the -c parameter when the SNMP agent is started. The default value is *public*.

OSASF
A value between 0 and 65535. There is no default. A value of 1 through 65535 indicates a port number and marks the corresponding TCP/IP instance as a candidate to communicate with OSA/SF for retrieval of SNMP management data regarding ATM devices and links. A value of 0 indicates that the corresponding TCP/IP instance is no longer a candidate to communicate with OSA/SF, in the event that the OSA/SF-to-TCP/IP connection is restarted.

When multiple TCP/IP instances specify that ATM management data retrieval is desired, it is recommended that all be configured with the same OSASF parameter. Only one TCP/IP instance connects directly to OSA/SF. Other instances connect to OSA/SF via this primary TCP/IP instance.

AGENT
A port number between 1 and 65535 used in establishing communication with the SNMP agent. For the TCP/IP SNMP subagent to communicate with the CS for OS/390 SNMP agent, the port number specified must match the port number specified (or defaulted) on the -p parameter when the SNMP agent is started. The default value is 161.

DISABLED
If specified in PROFILE.TCPIP at initialization, indicates that the SNMP subagent should not be started. Specify this keyword if no SNMP MIB objects

supported by the TCP/IP subagent will be requested. By default, the SNMP subagent is started by TCP/IP initialization.

If specified using obeyfile, indicates that the currently active subagent task should be terminated.

SNMP MIB objects supported by the CS for OS/390 SNMP agent and subagents other than the TCP/IP SNMP subagent will still be available. For information on which MIB objects are supported by the SNMP agent and subagent, see the*OS/390 SecureWay Communications Server: IP User's Guide*.

ENABLED
Indicates that the SNMP subagent should be started at the completion of this obeyfile or profile.

SETSENABLED
Indicates that the SNMP subagent should process SNMP 'set' requests. For example, SETSENABLED allows a user who issued an SNMP 'set' request, to cancel connections and start/stop devices using the SNMP agent security instead of RACF security.

SETSDISABLED
Indicates the SNMP subagent should not process SNMP 'set' requests.

ATMENABLED
Indicates that ATM Management support is required at this TCP/IP instance. For optimal performance, specify ATMENABLED only at the instance from which ATM Management support is needed. By default, ATM data retrieval is not enabled.

The SNMP subagent must be enabled, as it provides support for retrieval of SNMP management data about ATM devices and links. Therefore, do not specify the DISABLED parameter for this TCP/IP instance.

To retrieve ATM data, there must also be at least one TCP/IP active for which the OSASF parameter and its port number have been specified in the SACONFIG statement.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters. If you modify any parameters, other than DISABLED, you must recycle the subagent to affect the change.

Examples

```
SACONFIG COMMUNITY USACCESS AGENT 528
SACONFIG DISABLED
SACONFIG SETSENABLED ATMENABLED OSASF 2026
```

Usage Notes

- If DISABLED is specified, the subagent will *NOT* be started by TCP/IP initialization. If the SACONFIG statement itself is *NOT* specified, the subagent will be started (this is the default).
- The community string is case sensitive and must be 1 to 32 characters. It is not converted to uppercase by profile processing. It cannot contain any imbedded white space or control characters (such as blank, tab, end of line, or end of file) and cannot contain any imbedded semicolons (semicolons are treated as comment delimiters).

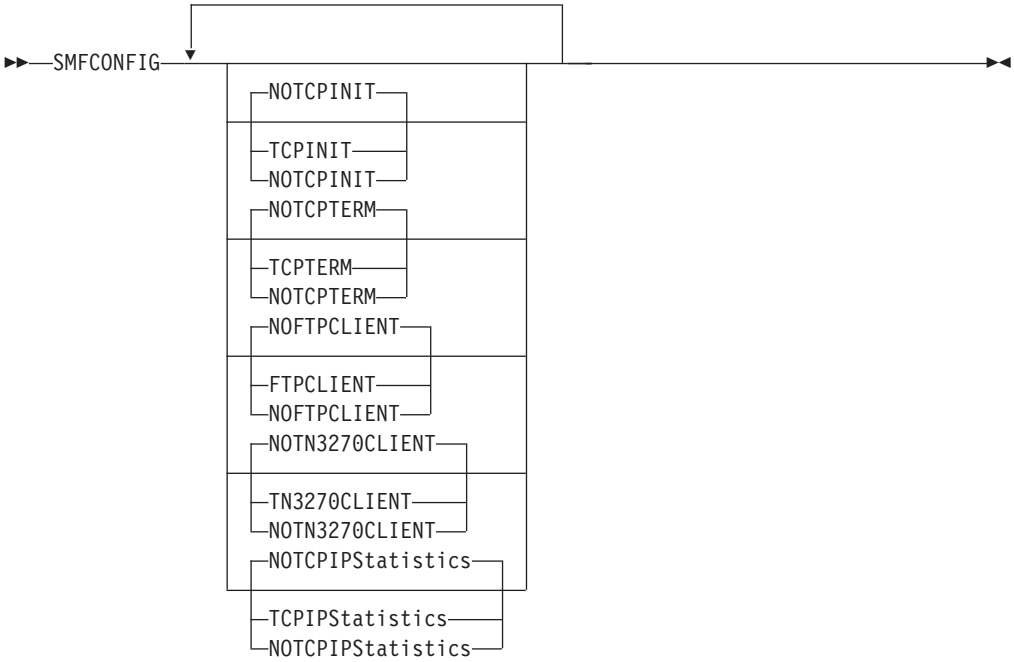
- At initialization time, the default for the SNMP subagent is to be enabled with 'set' support disabled.

SMFCONFIG Statement

Use the SMFCONFIG statement to provide SMF logging for Telnet, FTP, TCP API, and TCP stack activity.

Using SMFCONFIG to turn on SMF logging allows you to request that standard subtypes are assigned to the TCP/IP SMF records. The SMFPARMS statement provides a similar capability but requires the installation to select the subtype numbers to be used. Use of the SMFCONFIG parameter is recommended instead of SMFPARMS.

Syntax



Parameters

TCPINIT
Requests that SMF records of subtype 1 are created when TCP connections are established.

NOTCPINIT
Requests that no SMF records of subtype 1 are created when TCP connections are established.

TCPTERM
Requests that SMF records of subtype 2 are created when TCP connections are terminated.

NOTCPTERM
Requests that no SMF records of subtype 2 are created when TCP connections are terminated.

FTPCLIENT
Requests that SMF records of subtype 3 are created when a user invokes the FTP client command.

NOFTPCLIENT
Requests that no SMF records of subtype 3 are created when a user invokes the FTP client command.

TN3270CLIENT
Requests that SMF records of subtype 4 are created when a user invokes the Telnet client command.

NOTN3270CLIENT
Requests that no SMF records of subtype 4 are created when a user invokes the Telnet client command.

TCPIPSTATISTICS
Requests that SMF records of subtype 5 containing TCP/IP statistics are created. Note that these records are created periodically based on the SMF interval in effect.

Results of this parameter are printed in the SMF Parameters section of the output for the netstat (or onestat) command.

Note: This parameter is different from the ASSORTEDPARMS TCPIPSTATISTICS and GLOBALCONFIG TCPIPSTATISTICS parameters (see "ASSORTEDPARMS Statement" on page 128 and "GLOBALCONFIG Statement" on page 205). Results of those parameters are printed in the Global Configuration Information section of the output for the netstat (or onestat) command.

NOTCPIPSTATISTICS
Requests that no SMF records of subtype 5 are created.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example requests SMF records for TCP initialization, TCP termination, FTP client, Telnet client, and TCP/IP statistics:

```
SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
```

Usage Notes

- SMF must be active and properly configured. Because TCP/IP creates SMF records of type 118, SMF must be configured to allow for type 118 records to be created.
- Use of SMFCONFIG is preferable to SMFPARMS to standardize subtypes. If SMFPARMS is encountered after an SMFCONFIG statement, an error message is displayed and the SMFPARMS parameters are ignored. If SMFCONFIG is not coded, no SMF records are logged (assuming that SMFPARMS is not coded either).

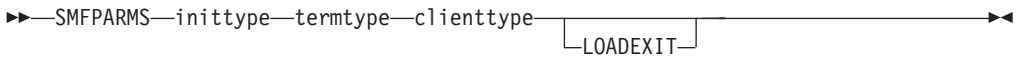
Related Topics

"Appendix B. SMF Records" on page 1163

SMFPARMS Statement

Use the SMFPARMS statement to log the use of TCP by applications using SMF log records. You can log Telnet and FTP client activity, and TCP API activity.

Syntax



Parameters

inittype
An integer in the range of 0 to 255 specifying the sub-type field in the API initialization records. The value 0 indicates that no API initialization will be written.

termtype
An integer in the range of 0 to 255 specifying the sub-type field in the API termination records. The value 0 indicates that no API termination records will be written.

clienttype
An integer in the range of 0 to 255 specifying the sub-type field in the FTP or Telnet client. The value 0 indicates that no FTP or Telnet client records will be written.

LOADEXIT
LOADEXIT is obsolete. A warning message is issued if it is used.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

- Either of the following statements would produce API initialization and termination records but no FTP or Telnet client records:

```
SMFPARMS    3   4 0
SMFPARMS    3   4
```

- The following statement would produce client records only:

```
SMFPARMS    0   0   5
```

- Because one of the parameters is missing, this statement would generate an error and not produce any records:

```
SMFPARMS  3
```

Usage Notes

- All parameters except LOADEXIT are required.
- The values for each sub-type should be unique.
- If *inittype*, *termtype*, or *clienttype* have the value of 0, no attempt will be made to write the respective record.
- The format of the log information differs for Telnet and FTP client activity, and TCP API activity.

Related Topics

- “Chapter 14. Configuring the File Transfer Protocol (FTP) Server” on page 523
- “Appendix B. SMF Records” on page 1163

SOMAXCONN Statement

Use the SOMAXCONN statement to specify a maximum length for the connection request queue created by the socket call listen().

Syntax

Parameters

maximum_queue_depth
The maximum number of pending connection requests queued for any listening socket. The default is 10.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example shows a SOMAXCONN statement specifying the default number of listening sockets.

```
SOMAXCONN                    10
```

Usage Notes

- This number is stored as a fullword integer, but most implementations of TCP/IP hardcode a value in the range of 5-10.
- This number is the maximum depth for any listening stream socket, but the programmer can specify a shorter queue length on each listen() socket call.
- There is a SOMAXCONN variable in the SOCKET.H file which is hardcoded at 10. If your C socket programs use this variable to determine what the acceptable maximum listen() backlog queue length is, remember to change the header file to reflect the value you specified for TCP/IP in SOMAXCONN maximum queue depth.

START Statement

Use the START statement to start a device that is currently stopped. This statement is usually specified at the end of *hlq*.PROFILE.TCPIP.

Syntax

Parameters

device_name
The name of the device to start. This should be the same *device_name* specified in the DEVICE statement.

Modifying

Modification is not applicable to this statement.

Examples

This example shows START statements that start devices LCS1 and LCS2.

```
START LCS1
START LCS2
```

Usage Notes

- VTAM must be active to START a device with TCP/IP.
- Each device to be started needs a separate START statement.
- The START statement can also be used in a VARY TCPIP command data set to start:
  - A newly-defined device
  - A device stopped with the STOP statement
  - A device that was never successfully started
- TCP/IP has a maximum of 255 started devices (that is, non-VIPA). However, if using OROUTED or OMPROUTE, the maximum number of interfaces is 255, and this maximum does not include VIPA interfaces. There is no maximum for static VIPA interfaces, but the maximum number of Dynamic VIPA interfaces is 64.
- For XCF connections, the device_name must be the cpname of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active on both nodes to start the device. If the connectivity is to a pre-R8 node, the XCF TRLE must also be active.
- The START statement is not valid for virtual devices. A virtual device is started automatically when a HOME entry is defined to it. It never leaves the started/active state.
- The START and STOP commands are processed *after* all other statements within the profile or obeyfile.
- For XCF connections, the device_name must be the cpname of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active to start the device.

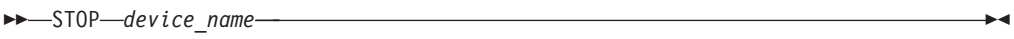
Related Topics

- “DEVICE and LINK Statements” on page 101
- “VARY Command—TCPIP Address Space” on page 264
- “STOP Statement” on page 246

STOP Statement

Use the STOP statement in a VARY TCPIP command data set to stop a device that is currently started.

Syntax



Parameters

device_name
The name of the device to be stopped. This should be the same *device_name* specified in the DEVICE statement.

Modifying

Modification is not applicable to this statement.

Examples

This example shows STOP statements that stop devices LCS1 and LCS2.

```
STOP LCS1
STOP LCS2
```

Usage Notes

- A virtual device can not be stopped.
- The START and STOP commands are processed *after* all other statements within the profile or obeyfile.
- The STOP statement can be used in a VARY TCPIP command data set to dynamically stop a device.

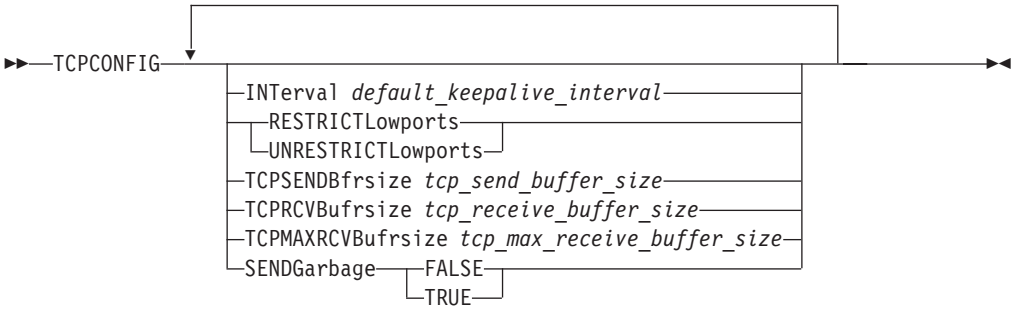
Related Topics

- “DEVICE and LINK Statements” on page 101
- “VARY Command—TCPIP Address Space” on page 264
- “START Statement” on page 244

TCPCONFIG Statement

Use the TCPCONFIG statement to update the TCP layer of TCP/IP.

Syntax



Parameters

INTERVAL *minutes*
The number of minutes TCP waits after receiving a packet for a connection before it sends a keep-alive packet for that connection. The range is from 0 to 35791 minutes, and the default is 120. A value of 0 will disable the keepalive function.

RESTRICTLOWPORTS
When set, ports 1 through 1023 are reserved for users by the PORT and PORTRANGE statements. The RESTRICTLOWPORTS parameter is confirmed by the message:

```
TCP ports 1 thru 1023 are reserved
```

Note: When RESTRICTLOWPORTS is specified, an application cannot obtain a port in the 0 through 1023 range unless it is explicitly reserved for that application, APF authorized, or it is not using the Pascal API and has an OMVS UID of zero. Therefore, be very careful in specifying this option. Some applications like RSH, LPR, NPF, and others, have a dependency on being able to obtain an available port in the 0 through 1023 range without having that port explicitly reserved for its use. **Use of this parameter is not recommended.**

UNRESTRICTLOWPORTS
When set, ports 1 through 1023 are not reserved. The UNRESTRICTLOWPORTS parameter is confirmed by the message:

```
TCP ports 1 thru 1023 are not reserved
```

TCPSENDBFRSIZE *tcp_send_buffer_size*
TCP send buffer size between 256 and 256K. The default is 16384 (16K). This value will be used as the default send buffer size for those applications which do not explicitly set the buffer size via SETSOCKOPT().

TCPRCVBUFRSIZE *tcp_receive_buffer_size*
TCP receive buffer size between 256 and TCPMAXRCVBUFRSIZE. The default is 16384 (16K). This value will be used as the default receive buffer size for those applications which do not explicitly set the buffer size via SETSOCKOPT().

TCPMAXRCVBUFRSIZE *tcp_max_receive_buffer_size*
The TCP maximum receive buffer size is the maximum value an application can set as its receive buffer size via SETSOCKOPT(). The minimum acceptable value is the value coded on TCPRCVBUFRSIZE, the maximum is 512K, and the default is 256K. If you do not have large bandwidth interfaces, you can use this parameter to limit the receive buffer size an application can set.

SENDGARBAGE
Specifies whether the keep-alive packets sent by TCP contain 1 byte of random data.

FALSE
Causes the packet to contain no data. This is the default.

TRUE
Causes the packet to contain 1 byte of random data and an incorrect sequence number, assuring that the data is not accepted by the remote TCP.

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example shows a TCPCONFIG statement that reserves ports 1 through 1023 for users by the PORT and PORTRANGE statements:

```
TCPCONFIG RESTRICTLOWPORTS
```

Usage Notes

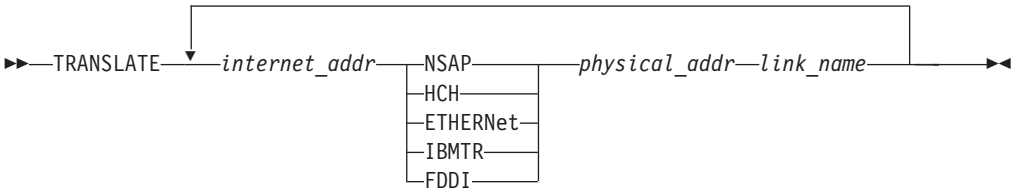
Some of these parameters can be specified on other statements (ASSORTEDPARMS, KEEPALIVEOPTIONS), and the settings from the last statement processed are used. For example, if RESTRICTLOWPORTS is not specified on ASSORTEDPARMS (and thus defaults to OFF) but is specified on a subsequent TCPCONFIG statement, RESTRICTLOWPORTS is set for TCP ports.

TRANSLATE Statement

Use the TRANSLATE statement to indicate the relationship between an internet address and the network address, on a specified link. You can use the TRANSLATE statement, with some limitations, for Ethernet, ATM, and token-ring hosts that do not support ARP.

The TRANSLATE statement is not valid for virtual devices, or point-to-point devices like CTC.

Syntax



Parameters

internet_addr
The internet address for which a translation is specified.

NSAP
Indicates the network address is an ATM address.

HCH
Indicates the network address is a HYPERchannel address.

ETHERNET
Indicates the network address is an Ethernet address.

IBMTR
Indicates the network address is a token-ring address.

FDDI
Indicates the network address is an FDDI address.

physical_addr
The network address corresponding to *internet_addr* and *link_name*. The format depends on the network type.

- For NSAP, specify a 40 digit hexadecimal value.

  Note: If the TRANSLATE statement is defining an ATM address for a TCP/IP stack on another OS/390 system, then the last hex digit of the 40-digit address cannot be 0 (0 is reserved for High Performance Routing (HPR) use by VTAM).

- For HCH, specify a 12-digit hexadecimal number of the form *ttxxxxxxhhcc*.

  tt The trunk mask. Use values other than FF only when advised to do so by Network Systems Corporation or by a HYPERchannel expert.

  xxxxxx
  These 6 digits are ignored.

  hh The remote adapter address.

  cc The meaning depends on the type of remote adapter. If the remote

adapter is attached to a VM TCP/IP or MVS TCP/IP system, then *cc* is the read port address (the lower of the two addresses that are attached to TCP/IP).

- For ETHERNET, IBMTR, and FDDI specify a 12-digit hexadecimal MAC address of the remote adapter.
  - For Ethernet, the remote host is assumed to use network headers DIX Ethernet format, not the 802.3 format.
  - For token ring, the translation table entry should not contain a token ring source routed bridge path.

link_name
A network link name (from the 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 a different *link_name*.

Modifying

To modify any values on the TRANSLATE statement, use a VARY TCPIP command with an OBEYFILE which contains a new TRANSLATE statement. All existing ARP entries will be deleted. To remove all static ARP entries from the ARP table, specify an empty TRANSLATE statement.

Notes:

1. If any HOME statement values were dynamically modified, all ARP static entries that correspond with the LINK names in the TRANSLATE statement will be deleted and replaced.
2. If any DEVICE/LINK statement values were dynamically deleted, all static ARP entries that correspond with the LINK names on the TRANSLATE statement will be deleted. In the OBEYFILE containing changed DEVICE/LINK statements and a new HOME statement, include a new TRANSLATE statement for the VARY TCPIP command.

See “VARY Command—TCPIP Address Space” on page 264 for more information about the VARY TCPIP commands.

Examples

This example shows the TRANSLATE statement for FDDI:

```
TRANSLATE
  9.67.51.3    FDDI    FF0000006702   FDDI1
  9.67.22.4    FDDI    FF0000009A05   FDDI1
```

Usage Notes

- Each configuration data set’s first executed TRANSLATE statement replaces the internal translation tables (the ARP table), including information dynamically added by ARP, with the new information. Subsequent TRANSLATE statements in the same data set add entries to the table.
- When an incorrect TRANSLATE statement entry is encountered, all entries following that entry on this TRANSLATE statement are ignored. Subsequent TRANSLATE statements in the same profile or obeyfile are processed.
- If the first TRANSLATE statement of a profile contains no internet address or link name, all addresses are removed from the TRANSLATE list.
- When using TRANSLATE to define the ATM address of a TCP/IP stack on another OS/390 system, it is important that the correct ATM address be specified. Start the ATM link on the other stack and use the onetstat -R ALL command to

display the ARP cache. The entry for the local IP address in this display will show exactly which ATM address must be specified in the TRANSLATE statement on the first stack.

- Implementations of OSPF that listen for an address other than the 0xc000.0004.0000 at the DLC layer will not be able to communicate with the TCP/IP stack unless they code the following TRANSLATE statement in the TCP profile:

  ```
  TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFFFF linkname
  ```

- Some token ring hardware does not recognize the RFC 1469 mandated functional MAC address for multicast. The TRANSLATE statement can be used to configure a token ring link to broadcast multicast datagrams as an alternative to using the functional MAC address. Use the reserved class D address 224.0.0.0 with one of the following special physical addresses:
  - FFFFFFFFFFFF for all rings broadcast.
  - C00000040000 to reset back to the default functional address.

  The following are examples of how to specify each method:
  - All rings:

    ```
    TRANSLATE
      224.0.0.0  IBMTR  FFFFFFFFFFFF  linkname
    ```

  - Assigned functional address:

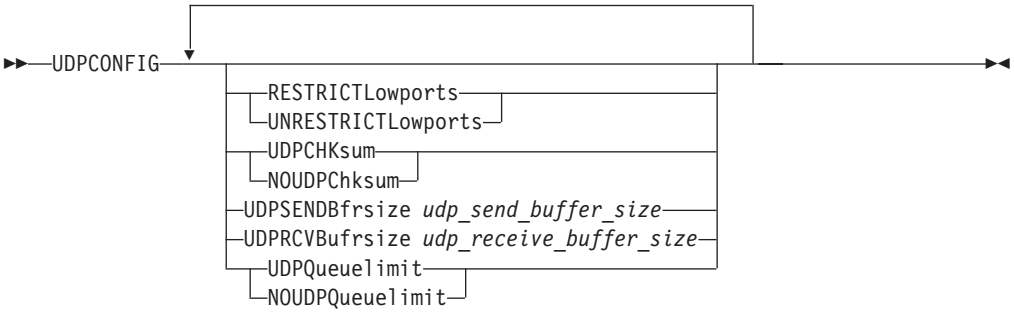
    ```
    TRANSLATE
      224.0.0.0  IBMTR  C00000040000  linkname
    ```

  Note: The TRANSLATE statement is effective on a per link basis. You do not have to code a TRANSLATE statement if you want the assigned functional address, as it is the default method.

UDPCONFIG Statement

Use the UDPCONFIG statement to update the UDP layer of TCP/IP.

Syntax



Parameters

RESTRICTLOWPORTS
When set, ports 1 through 1023 are reserved for users by the PORT and PORTRANGE statements. The RESTRICTLOWPORTS parameter is confirmed by the message:

```
UDP ports 1 thru 1023 are reserved
```

Note: When RESTRICTLOWPORTS is specified, an application cannot obtain a port in the 0 through 1023 range unless it is explicitly reserved for that application. Therefore, be very careful in specifying this option. Some UDP applications might have a dependency on being able to obtain an available port in the 0 through 1023 range without having that port explicitly reserved for its use. **Use of this parameter is not recommended.**

UNRESTRICTLOWPORTS
Ports 1 through 1023 are not reserved. The UNRESTRICTLOWPORTS parameter is confirmed by the message:

```
UDP ports 1 thru 1023 are not reserved
```

UDPCHKSUM
Used to ensure UDP does check summing.

NOUDPCHKSUM
Used to ensure UDP does not do check summing.

UDPSENDBFRSIZE *udp_send_buffer_size*
Set UDP send buffer size.

UDPRCVBUFRSIZE *udp_receive_buffer_size*
Set UDP receive buffer size.

UDPQUEUELIMIT
Used to set a queue limit for UDP. The UDPQUEUELIMIT parameter is confirmed by the message:

```
A limit on incoming UDP datagram queue set
```

NOUDPQUEUELIMIT
Used to specify that UDP should not have a queue limit. The NOUDPQUEUELIMIT parameter is confirmed by the message:

```
No limit on incoming UDP datagram queue set
```

Modifying

To modify parameters for this statement, you must respecify the statement with the new parameters.

Examples

This example shows a UDPCONFIG statement that uses check summing, sets no queue limit, and sets the send buffer size to 8192:

```
UDPCONFIG UDPCHK NOUDPQ UDPSENDB 8192
```

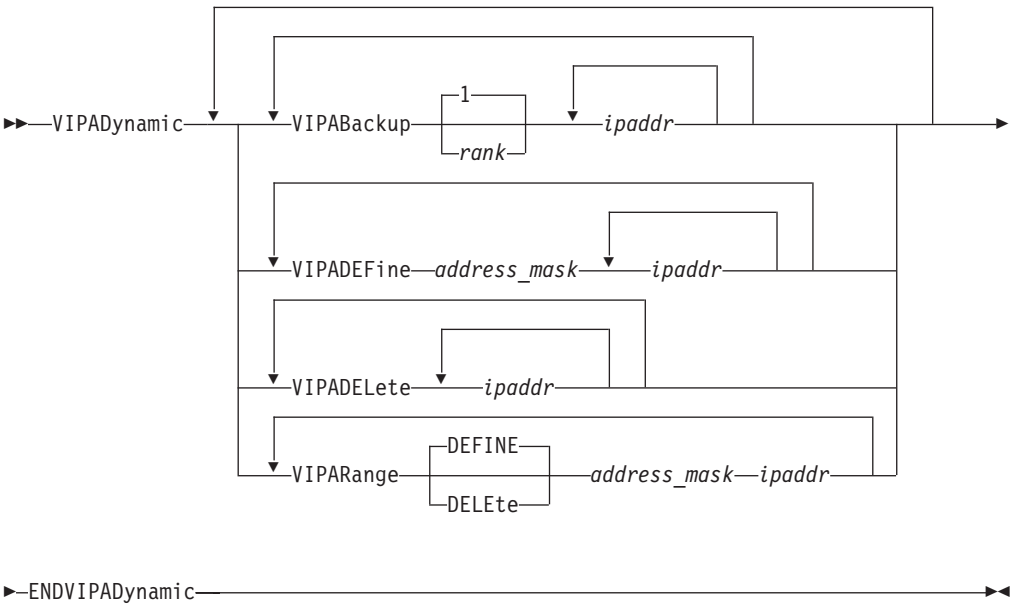
Usage Notes

RESTRICTLOWPORTS can be specified on the ASSORTEDPARMS statement, and the settings from the last statement processed are used. For example, if RESTRICTLOWPORTS is not specified on ASSORTEDPARMS (and thus defaults to OFF) but is specified on a subsequent UDPCONFIG statement, RESTRICTLOWPORTS is set.

VIPADYNAMIC Statement

Use the VIPADYNAMIC statement to start a block of definitions related to Dynamic VIPAs.

Syntax



Parameters

VIPABACKUP

Designates one or more Dynamic VIPAs for which this stack will provide automatic backup if the owning stack fails. Another stack is expected, but not required, to have this same VIPA defined with a VIPADYNAMIC VIPADEFINE statement.

rank

Specifies the intended order of the VIPAs in this VIPABACKUP list in their respective backup chains, relative to other stacks in those backup chains. Larger numerical rank values move the respective stacks closer to the beginning of the backup chain.

rank can be set to any integer from **1** (end of the backup chain) through **254** (start of the backup chain). Values **0** and **255** are reserved for use by the stacks themselves to temporarily force stack entries to the start or the end of the backup chain until an expected transition takes place.

The default is a rank of **1**.

ipaddr

Specifies the specific VIPA to be backed up. More than one *ipaddr* can be specified on a single VIPABACKUP. All *ipaddrs* specified on a single VIPABACKUP will have the same rank. Use multiple VIPABACKUP addresses to define different ranks for different *ipaddrs*.

VIPADEFINE

Designates one or more Dynamic VIPAs which this stack should initially own and support. Other stacks may provide backup for these VIPAs if this stack fails.

address_mask

Provides the net mask that determines how many of the bits of the IP address determine the net. All IP addresses in the same VIPADEFINE list must belong to the same net. That is, if the *address_mask* is logically ANDed with all the *ipaddr*s in the list, the resulting values must all be the same. The first IP address in the list determines the NAddress.

This value must meet the normal mask definition rules:

1. When converted to binary, the most significant bit must be a **1**.
2. When converted to binary, all bits less significant than (to the right of) the first **0** encountered must also be **0**.

In other words, the IP addresses in the subnet must be a single contiguous range of IP addresses.

ipaddr

Specifies the specific VIPA to be defined. More than one *ipaddr* can be specified on a single VIPADEFINE.

If a VIPA in this VIPADEFINE list is already active on another stack as a Dynamic VIPA that was activated by VIPADEFINE or VIPABACKUP, the other stack gives the VIPA to this stack only when there are no active connections to that VIPA on the other stack. If two or more stacks in the sysplex have the same VIPA in VIPADYNAMIC VIPADEFINE statements, with different address masks, the stack that ends up with the active VIPA determines the address mask.

If a VIPA in this VIPADEFINE list is already active on this stack or another stack either as an IP address in a HOME statement or as a Dynamic VIPA activated via an IOCTL or a BIND implicit activation, the VIPA in the VIPADEFINE list is rejected and an error message is issued.

VIPADELETE

Removes one or more Dynamic VIPAs from the VIPADEFINE or VIPABACKUP list in which each occurs. You can also use VIPADELETE to delete a VIPA that was established by implicit request via a BIND to a specific address, or via a SIOCSVIPA IOCTL .

ipaddr

Specifies the specific VIPA to be deleted from the stack. More than one *ipaddr* can be specified on a single VIPADELETE.

VIPARANGE

Defines a net in which requests for activating a Dynamic VIPA, via a BIND or SIOCSVIPA IOCTL, will be honored; or removes the definition for such a net.

Note: VIPARANGE definitions that are common to more than one stack should be defined in a common file and included in the appropriate stack profiles. This can help you avoid keying errors that could result in a failure to activate an application on a stack.

DEFINE

Specifies that this definition is to be added to the list of defined VIPARANGEs. DEFINE is the default.

DELETE
Specifies that this net (with the same *address_mask* and NAddress) is to be removed from the list of allowable ranges for IOCTL or BIND implicit Dynamic VIPA activation.

address_mask
Provides the net mask that determines how many of the bits of the IP address determine the net.

This value must meet the normal mask definition rules:
1. When converted to binary, the most significant bit must be a **1**.
2. When converted to binary, all bits less significant than (to the right of) the first **0** encountered must also be **0**.

In other words, the IP addresses in the subnet must be a single contiguous range of IP addresses.

ipaddr
Determines a VIPARANGE net value when ANDed with the specified address mask. Any Dynamic VIPA requested via IOCTL or by implicit BIND to a specific address must match some defined VIPARANGE net value, after the Dynamic VIPA has been logically ANDed with the corresponding address mask.

Modifying

- To modify the VIPADEFINE statement, follow these steps:
  1. To remove one or more of the IP addresses, use:
     ```
     VIPADELETE  ipaddr  [ipaddr ...]
     ```
  2. To change the mask for one or more of the IP addresses, use:
     ```
     VIPADELETE  ipaddr  [ipaddr ...]
     VIPADEFINE  new_mask  ipaddr  [ipaddr ...]
     ```
     Note: If ipaddr is currently active, VIPADELETE will break any existing connections and cause the Dynamic VIPA to be activated elsewhere in the sysplex if there is another stack prepared to activate it.
  3. To modify the VIPABACKUP to remove the ipaddr as a Dynamic VIPA backup, use:
     ```
     VIPADELETE ipaddr
     ```
  4. To change the rank (if the ipaddr is not currently active on this stack) use:
     ```
     VIPABACKUP  new_rank ipadd
     ```
     However, if the ipaddr is currently active, you must first delete it and then configure it with the new rank by using:
     ```
     VIPADELETE   ipaddr
     ```
     Note: If ipaddr is currently active, VIPADELETE will break any existing connections and cause the Dynamic VIPA to activate elsewhere in the sysplex if there is another stack prepared to activate it.
  5. To modify the VIPARANGE to remove a VIPARANGE, use:
     ```
     VIPARANGE DELETE mask ipaddr
     ```
  6. To change the subnet for a VIPARANGE, you can replace the subnet by using:

```
VIPARANGE DELETE original_mask original_ipaddr
VIPARANGE new_mask new_ipaddr
```

Alternatively, you can enlarge the subnet by using:

```
VIPARANGE  mask2  ipaddr2
```

This configures a VIPARANGE where Mask2 ANDed with ipaddr2 determine a subnet that overlaps or includes the original one.

- VIPADELETE is used only to remove an existing Dynamic VIPA from the configuration. Modification is not applicable.

Examples

This example shows the use of the VIPADEFINE and VIPABACKUP statements with a VIPADYNAMIC/ENDVIPADYNAMIC block.

```
VIPADYNAMIC
   VIPADEFINE  255.255.255.192  201.2.10.11  201.2.10.12
   VIPABACKUP  100              201.2.10.13
ENDVIPADYNAMIC
```

Usage Notes

A given stack is limited to no more than 64 configured BackupDynamic VIPAs. A *configured* Dynamic VIPA is one that is defined and activated in any of the following ways:

1. Via VIPADEFINE, and kept since
2. Via VIPABACKUP, and the VIPA was obtained because the owning stack failed or deleted the VIPA, and this stack was first in the backup list
3. Via an IOCTL SIOCSVIPA DEFINE
4. Via a BIND to a specific IP address that was not in use elsewhere in the sysplex, and to which this stack had a covering VIPARANGE

Chapter 5. Operator Commands and System Administration

This chapter includes information about operator commands and system administration information. Topics include:

- Table of commands
- Administration overview
- Starting and stopping TCP/IP servers
- Operator commands

Note: Because CS for OS/390 runs in a variety of environments, some commands have several implementations. Those variations of standard commands (such as START) are discussed in the chapters about the particular servers. For more details about commands, you can also see *OS/390 SecureWay Communications Server: IP User's Guide*.

Table of Commands

Use Table 11 to help you locate information about some the commands described in this book. In some cases, this information includes a syntax diagram or other configuration information. This table is not a comprehensive list of all the available commands, but a tool designed to help direct you to information about some frequently used commands.

Table 11. Table of Commands

Command	Purpose		Additional information
MVS Environment			
START	Use START to dynamically start a TCP/IP server or address space (including the TCP/IP address space).		"START Command" on page 261
STOP	Use STOP to stop a TCP/IP server or address space (including the TCP/IP address space) that is in execution.		"STOP Command" on page 261
MODIFY	Use MODIFY to dynamically change the characteristics of an active task.		"Modifying Server Parameters" on page 262
VARY	Use VARY TCPIP to control some functions of the TCP/IP address space from the operator's console.		"VARY Command—TCPIP Address Space" on page 264
DISPLAY	Use DISPLAY TCPIP from the MVS operator's console to display the syntax of MVS operator commands, or to receive NETSTAT, TELNET, OMPROUTE, or SYSPLEX information.		"DISPLAY Command—TCPIP Address Space" on page 273
TSO Environment			
CONVXLAT	Use CONVXLAT to customize the binary translation table codefiles for 3270 DBCS transform mode.		"Chapter 36. Using Translation Tables" on page 1133
LOGON	TSO/Telnet	Use the logon command to request a session with an application program.	"LOGON Command" on page 477

Table 11. Table of Commands (continued)

Command	Purpose		Additional information
LOGOFF	TSO/Telnet	Use the logoff command to drop the TCP/IP connection between the server and the client.	"LOGOFF Command" on page 478
IBMTEST	TSO/Telnet	Use the ibmtest command to verify that the client is connected to Telnet.	"IBMTEST Command" on page 479
SMSG	TSO/LPD	Use the SMSG command provides an interface to the LPD address space.	"Operating the LPD Server Using the SMSG Interface" on page 916
EXT@SRVT	TSO/Kerberos	Use EXT@SRVT to generate a key data set for instances.	"EXT@SRVT Command" on page 883
KADMIN	TSO/Kerberos	Use KADMIN to add, get, or modify a Kerberos database.	"KADMIN Command" on page 884
KDB@DEST	TSO/Kerberos	Use KDB@DEST to erase the ADM@SRV.PRINCPL.DAT and ADM@SRV.PRINCPL.IDX data sets.	"KDB@DEST Command" on page 886
KDB@EDIT	TSO/Kerberos	Use KDB@EDIT to register users and services to the Kerberos database.	"KDB@EDIT Command" on page 887
KDB@INIT	TSO/Kerberos	Use KDB@INIT to create and format the Kerberos database.	"KDB@INIT Command" on page 889
KSTASH	TSO/Kerberos	Use KSTASH to create the ADM@SRV.ETC.K data set.	"KSTASH Command" on page 891
MAKESITE	TSO/TCP/IP	Use to create the hlq.HOSTS.SITEINFO and hlq.HOSTS.ADDRINFO data sets.	"MAKESITE Command" on page 313
TESTSITE	TSO/TCP/IP	Use to verify the hlq.HOSTS.SITEINFO and hlq.HOSTS.ADDRINFO data sets.	"TESTSITE Command" on page 316
HOMETEST	TSO/TCP/IP	Use to verify your host name and address configuration.	"Verify Host Name and Address Configuration" on page 41
SMTPNJE	TSO/SMTP	Use to create the user_id.SMTPNJE.HOSTINFO data set.	"Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)" on page 647
Shell Environment			
onslookup	OS/390 shell, DNS	Use onslookup to query the name server.	"Querying Name Servers" on page 718
nsupdate	OS/390 shell, DNS	Use nsupdate on the server to create key entries to the /etc/ddns.dat file for DHCP supported clients.	"Defining DDNS Key Files for the A Record" on page 777

Administration Overview

After your TCP/IP system is configured, you can use these MVS commands to dynamically start, stop, and control the servers:

- MODIFY
- START
- STOP
- DISPLAY TCPIP
- VARY TCPIP

All of the servers or address spaces except PORTCPRC support START and STOP. PORTCPRC supports START but does not support STOP.

Recommendation:

Although the MVS commands can accept *procname.identifier* to specify the server or address space, the AUTOLOG statement in *hlq*.PROFILE.TCPIP ignores the *identifier* portion. Therefore, it is preferred that you use the member name of the cataloged procedure on the AUTOLOG statements in *hlq*.PROFILE.TCPIP.

Starting and Stopping TCP/IP Servers

Any of the servers you specify on the AUTOLOG statement in the PROFILE.TCPIP data set will start automatically when you start the TCPIP address space. You can use the STOP command to dynamically stop an individual server or address space (including the TCP/IP address space) and the START command to restart it.

START Command

Use the START command to dynamically start a TCP/IP server or address space (including the TCP/IP address space).

procname
The name of a member in a cataloged procedure library. For the servers, this should be the same name specified on the PORT statement in the PROFILE.TCPIP data set.

STOP Command

Use the STOP command to stop a TCP/IP server or address space (including the TCP/IP address space) that is in execution.

When you issue the STOP command, the following sequence of events occurs, depending on whether connected servers have outstanding calls to TCP/IP.

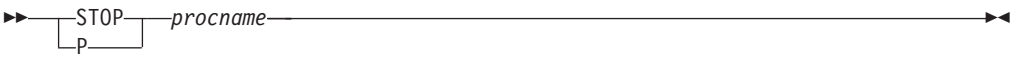
For each server with outstanding calls to TCP/IP

The TCP/IP address space notifies the server that TCP/IP is coming down and requests that the server terminate normally.

If the server does not terminate normally, TCP/IP causes the server to abend with abend code 422. The abend does not appear in a dump; however it is recorded in the SYS1.LOGREC data set. The outstanding socket call receives error number 1041 EIBMBADPOSTCODE.

For each connected server that does not have outstanding calls

The TCP/IP address space notifies the server that TCP/IP is coming down and drives the server's asynchronous error exit routine, if there is one.



procname
The name of the procedure you want to stop. This should be the same member name used to start the server, either on the START command or the AUTOLOG statement in the PROFILE.TCPIP data set.

Modifying Server Parameters

The MODIFY command allows you to dynamically change the characteristics of an active task.

Modify Command

This is the general format of MODIFY:

procname
The name of the member in a procedure library that was used to start the server or address space.

parameter
Any of the parameters that are valid for the server.

The following servers or address spaces support the MVS MODIFY command. Not all servers support the same parameters. Further descriptions of the supported parameters are in the chapter for that server as indicated in Table 12.

Table 12. Servers or Address Spaces That Support the MVS Modify Command

Server/Addr Space	Main Parameters	Additional Information
FTP server	NODUMP, TRACE, NOTRACE, JTRACE, NOJTRACE, DUMP, JDUMP, NOJDUMP, UTRACE, NOUTRACE,	"Starting, Stopping, and Tracing the FTP Server" on page 610
SNALINK LU0	HALT, PKTTRACE	"MODIFY Command—SNALINK LU0" on page 343

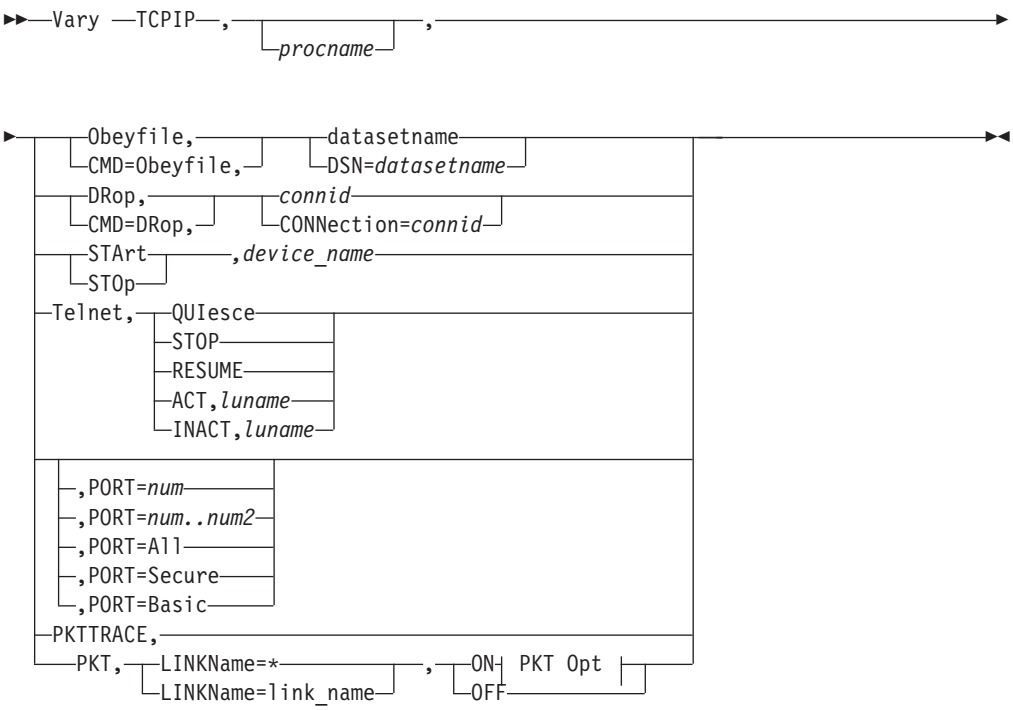
Table 12. Servers or Address Spaces That Support the MVS Modify Command (continued)

Server/Addr Space	Main Parameters	Additional Information
SNALINK LU6.2	CANCEL, DROP, HALT, LIST, PKTTRACE, RESTART, TRACE	"MODIFY Command—SNALINK LU 6.2" on page 356
X.25 NPSI server	CANCEL, DEBUG, EVENTS, HALT, LIST, PKTTRACE, RESTART, SNAP, TRACE, TRAFFIC	"Operating the X.25 NPSI Server Using the MODIFY Command" on page 382
REXEC	EXIT, TSOPROC, MSGCLASS, TSCLASS, TRACE	"MODIFY Command—Remote Execution Server" on page 633
RouteD server	PARMS, GATEWAYS, TABLES, RECONFIG, PROFILE	"Controlling OROUTED with the MODIFY Command" on page 945
OMPROUTE	KILL, RECONFIG, ROUTESA, ISPF, TRACE, DEBUG, SADEBUG	"Controlling OMPROUTE with the MODIFY Command" on page 958
NCPROUTE server	C, PARMS, PROFILE, QUERY, GATEWAYS, TABLES	"Controlling the NCPROUTE Address Space with the MODIFY Command" on page 1061
SMNP Agent	INTERVAL, TRACE	*OS/390 SecureWay Communications Server: IP User's Guide*
SLA Subagent	TRACE, QUERY	"Controlling the SLA Subagent with the Modify Command" on page 1119
TNF	DISPLAY, REMOVE	"Step 3: Configuring VMCF and TNF" on page 37
VMCF	DISPLAY, REMOVE	"Step 3: Configuring VMCF and TNF" on page 37

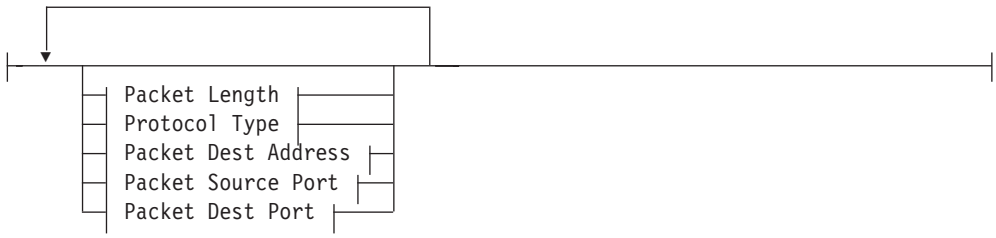
VARY Command—TCPIP Address Space

Use the VARY TCPIP command to control some functions of the TCP/IP address space from the operator's console. Listed below is the complete syntax for the VARY command. The remaining sections of this chapter separate the various forms of the VARY command by function and include parameter descriptions, examples, and usage notes.

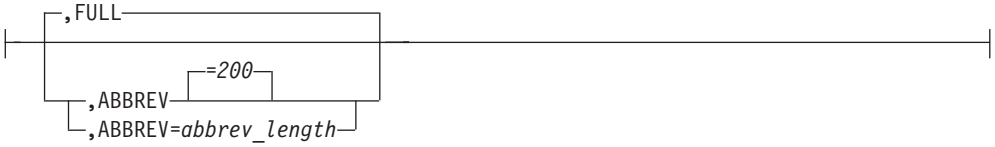
Syntax



PKT Opt:

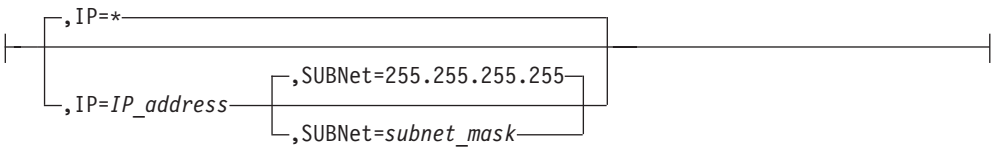


Packet Length:



Protocol Type:

Packet Dest Address:



Packet Source Port:

Packet Dest Port:

Security Considerations for the VARY Command

You can restrict access to the VARY TCPIP command by defining RACF profiles under the OPERCMDS class and specifying the list of users that are authorized to issue the VARY TCPIP command. You can decide on the level of control that is appropriate for your installation. For example, you may want to allow a user to be able to start or stop a TCP/IP device using the VARY TCPIP command but you do not want the user to be able to modify the TCP/IP configuration.

The RACF profile names that restrict access to each of the VARY TCPIP commands are listed under the ″Usage Notes″ in the section that describes these commands. Note that CONTROL access to each profile is required for a user to be able to issue the VARY TCPIP command.

To restrict all of the VARY TCPIP commands, you can define a generic profile as follows:

```
RDEFINE OPERCDS (MVS.VARY.TCPIP.**) UACC(NONE)
PERMIT MVS.VARY.TCPIP.** ACCESS(CONTROL) CLASS(OPERCMDS)
   ID(USER1)
```

In this example, only user ID USER1 is allowed to issue any VARY TCPIP operator commands. In another example, if you wanted to restrict usage of the VARY TCPIP,,OBEYFILE command to user ID USER2 you could make the following definitions:

```
RDEFINE OPERMCDS (MVS.VARY.TCPIP.OBEYFILE) UACC(NONE)
PERMIT (MVS.VARY.TCPIP.OBEYFILE) ACCESS(CONTROL)
   CLASS(OPERCMDS) ID(USER2)
```

Note that for any of these profiles to take a effect, the RACF OPERCMDS class must be activated. If you desire to have the ability to define generic RACF profiles for these profiles you also need to ensure that the appropriate RACF options are specified. This can be accomplished by the following RACF commands:

```
SETR CLASSACT(OPERCMDS)
SETR GENERIC(OPERCMDS)
SETR GENCMD(OPERCMDS)
SETR RACLIST(OPERCMDS)
```

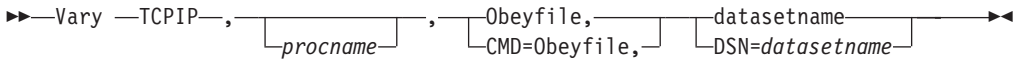
Also note that before the profiles take effect, a refresh of these RACF profiles may be required. This can be accomplished by the following RACF commands:

```
SETR GENERIC(OPERCMDS) REFRESH
SETR RACLIST(OPERCMDS) REFRESH
```

VARY TCPIP,,OBEYFILE

Use the VARY TCPIP,,OBEYFILE command to make temporary dynamic changes to the system operation and network configuration without stopping and restarting the TCP/IP address space.

Syntax



Parameters

procname
The identifier of the TCP/IP address space. When the *procname* parameter is not specified, there can be only one TCP/IP address space started. If more than one TCP/IP address space is available and no *procname* is specified, the request will fail with an error message.

CMD=OBEYFILE or OBEYFILE
Specify this parameter to make temporary dynamic changes to the system operation and network configuration without stopping and restarting the TCP/IP address space. These changes are in effect until the TCPIP cataloged procedure is started again or until another VARY OBEYFILE overrides them. Put your changes in the data set specified by the parameter *datasetname*. You can maintain different data sets that contain a subset of the TCP/IP configuration statements and activate them while TCP/IP is running.

DSN=*datasetname* **or** *datasetname*
The value *datasetname* is required after specifying the OBEYFILE parameter. *datasetname* is the name of a data set containing TCPIP configuration statements. *datasetname* must be a cataloged data set and specified as fully qualified without any quotes. *datasetname* can be either a sequential data set or a member in a PDS.

Examples

Following are examples of updating system operation and network configuration information without stopping and restarting the TCP/IP address space.

1. The first example is directed to a TCPIP address space started by the identifier TCPPROC, and assumes the sequential data set USER99.TCPIP.OBEYFIL1 contains TCP/IP configuration statements:

   ```
   VARY TCPIP,TCPPROC,CMD=OBEYFILE,DSN=USER99.TCPIP.OBEYFIL1
   ```

2. The next example assumes there is only one TCPIP address space and that OBEYFIL2 is a member of the PDS USER99.TCPIP and contains TCP/IP configuration statements:

   ```
   VARY TCPIP,,O,USER99.TCPIP(OBEYFIL2)
   ```

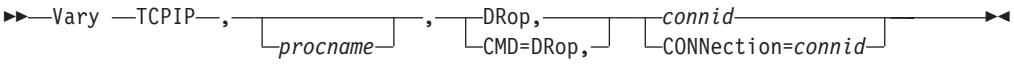
Usage Notes

1. Authorization is through the user's RACF profile containing the MVS.VARY.TCPIP.OBEYFILE definition for CMD=OBEYFILE.
2. The DSN= parameter cannot be an HFS file.

VARY TCPIP,,DROP

Use the VARY TCPIP,,DROP command to drop a connection.

Syntax



Parameters

procname
The identifier of the TCP/IP address space. When the *procname* parameter is not specified, there can be only one TCP/IP address space started. If more than one TCP/IP address space is available and no *procname* is specified, the request will fail with an error message.

CMD=DROP or DROP
Specify this parameter to drop a connection.

CONNECTION=*connid* **or** *connid*
The value *connid* is required after specifying the DROP parameter. *connid* is the connection identifier for the TCPIP socket connection that is to be dropped. Issue the onetstat -c command, the D TCPIP,,NETSTAT,CONN command, or the TSO NETSTAT CONN command to obtain the connection identifier for the TCP/IP socket connection that you want to drop.

Examples

Following are examples of dropping TCP/IP socket connections.

- The first example is directed to a TCPIP address space started by the identifier TCPPROC and demonstrates how to drop a TCP connection number 5001:

  ```
  VARY TCPIP,TCPPROC,CMD=DROP,CONNECTION=5001
  ```

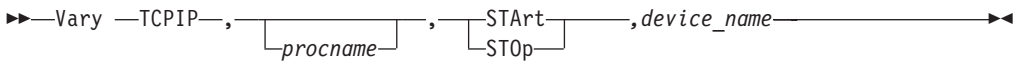
- The next example assumes there is only one TCPIP address space and demonstrates how to drop a UDP connection number 6001:

  ```
  VARY TCPIP,,CMD=DROP,CONNECTION=6001
  ```

VARY TCPIP to START or STOP a Device

Use the VARY TCPIP,,START command to start a device and the VARY TCPIP,,STOP command to stop a device.

Syntax



Parameters

procname
The identifier of the TCP/IP address space. When the *procname* parameter is not specified, there can be only one TCP/IP address space started. If more than one TCP/IP address space is available and no *procname* is specified, the request will fail with an error message.

STArt
Start a device known to TCP/IP.

STOp
Stop a device known to TCP/IP.

device_name
The name of the device to be started or stopped.

Examples

Following is an example of starting a device:

```
V TCPIP,,START,DEVD00
EZZ0060I PROCESSING COMMAND: VARY TCPIP,,START,DEVD00
```

Usage Notes

- Authorization is through the user's RACF profile containing the MVS.VARY.TCPIP.STRTSTOP definition for START or STOP.
- When the VARY START command is used for XCF connection (specifying the CP name of the other node), the ISTLSXCF major node must be active on both nodes and the XCF TRLE for the connection must be active.

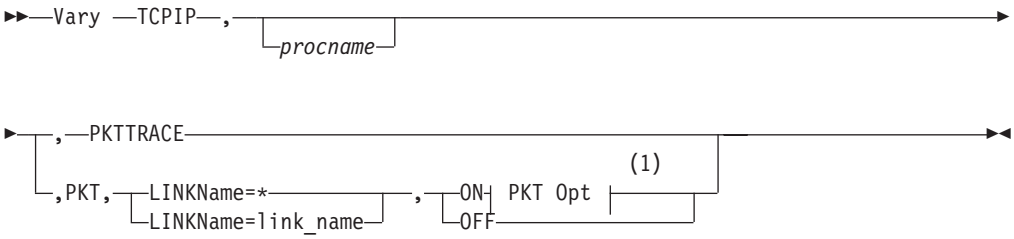
VARY TCPIP,,TELNET

Use the VARY TCPIP,,TELNET command to control TELNET. For details, see “Operating the Telnet Server Using the VARY TCPIP Command Set” on page 489.

VARY TCPIP,,PKTTRACE

Use the VARY TCPIP,,PKTTRACE command to set up tracing.

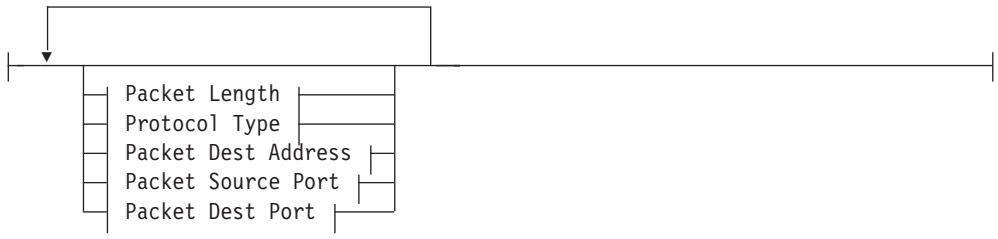
Syntax



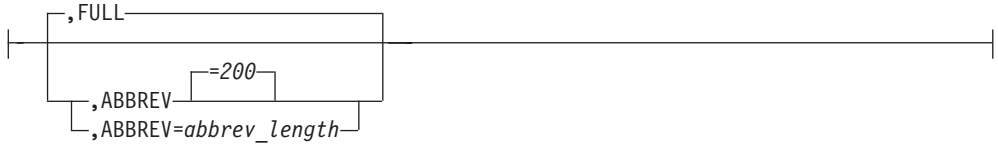
Notes:

1 Each option can be specified only once. The order of options is not important.

PKT Opt:

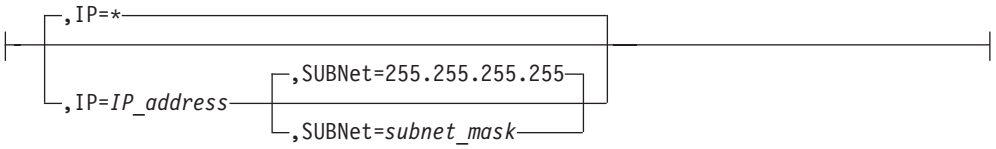


Packet Length:



Protocol Type:

Packet Dest Address:



Packet Source Port:

Packet Dest Port:

Parameters

procname
The identifier of the TCP/IP address space. When the *procname* parameter is not specified, there can be only one TCP/IP address space started. If more than one TCP/IP address space is available and no *procname* is specified, the request will fail with an error message.

PKTTRACE, PKT
Specifies this command is for PKTTRACE information.

ON
Turns PKTTRACE on. For descriptions of the PKTTRACE options, see "PKTTRACE Statement" on page 223.

OFF
Turns PKTTRACE off.

Usage Notes

Authorization is through the user's RACF profile containing the MVS.VARY.TCPIP.PKTTRACE definition for PKTTRACE.

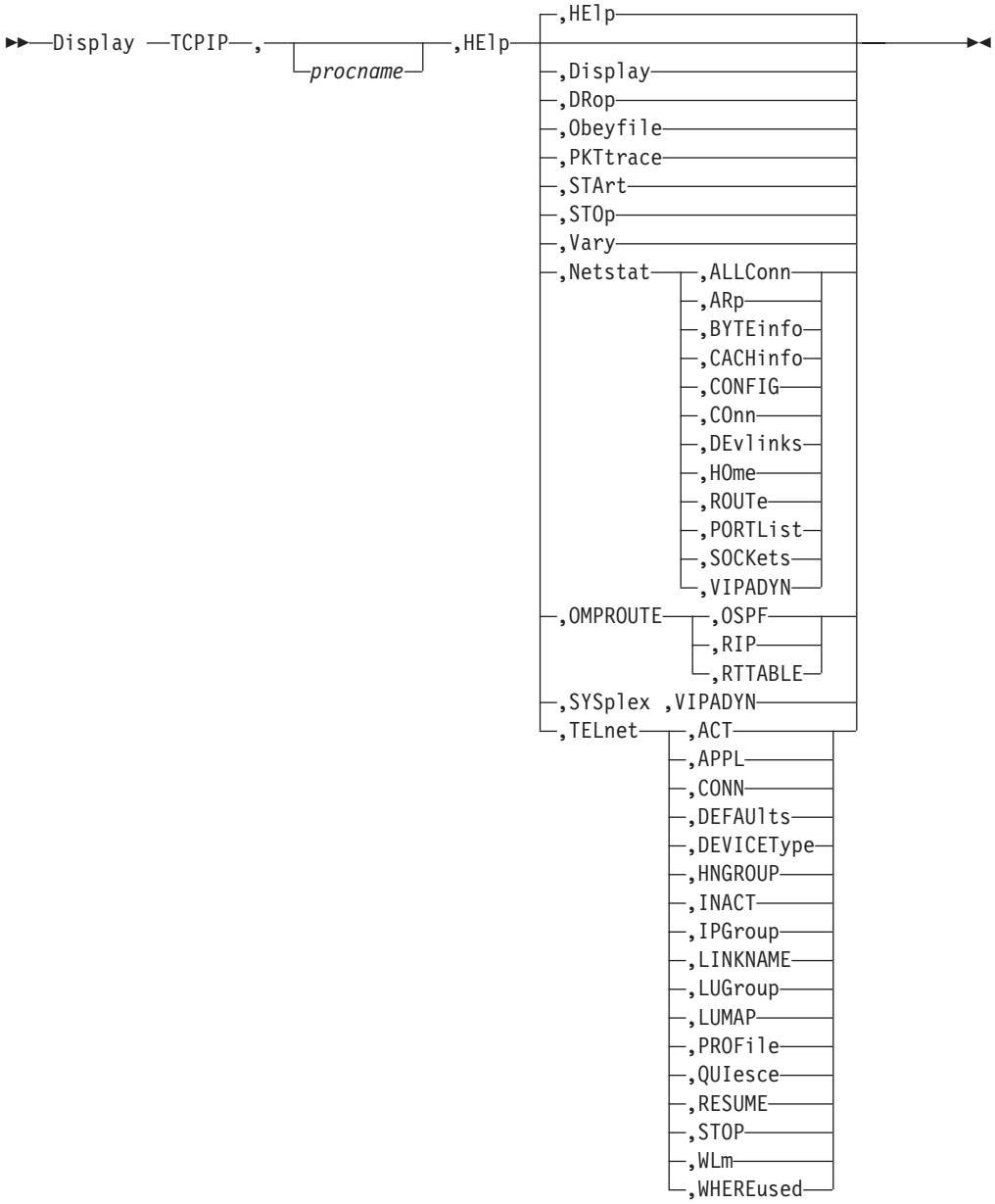
DISPLAY Command—TCPIP Address Space

Use the DISPLAY TCPIP command from the MVS operator's console to display the syntax of MVS operator commands, or to receive NETSTAT or TELNET information.

DISPLAY TCPIP,,HELP

Use the DISPLAY TCPIP,,HELP command from the MVS operator's console to display the syntax of MVS operator commands.

Syntax



Parameters

HELP
Show help on the DISPLAY HELP command.

DISPLAY
Show help on the DISPLAY TCPIP command.

DROP
Show help on the VARY DROP command.

OBEYFILE
Show help on the VARY OBEYFILE command.

PKTTRACE
Show help on the DISPLAY PKTTRACE command.

START
Show help on the DISPLAY START command.

STOP
Show help on the DISPLAY STOP command.

VARY
Show help on the VARY TCPIP command.

NETSTAT
Show help on the DISPLAY NETSTAT command.

ALLCONN
Show help on the DISPLAY NETSTAT,ALLCONN command.

ARP
Show help on the DISPLAY NETSTAT,ARP command.

BYTEINFO
Show help on the DISPLAY NETSTAT,BYTEINFO command.

CACHINFO
Show help on the DISPLAY NETSTAT,CACHINFO command.

CONFIG
Show help on the DISPLAY NETSTAT,CONFIG command.

CONN
Show help on the DISPLAY NETSTAT,CONN command.

DEVLINKS
Show help on the DISPLAY NETSTAT,DEVLINKS command.

HOME
Show help on the DISPLAY NETSTAT,HOME command.

ROUTE
Show help on the DISPLAY NETSTAT,ROUTE command.

PORTLIST
Show help on the DISPLAY NETSTAT,PORTLIST command.

SOCKETS
Show help on the DISPLAY NETSTAT,SOCKETS command.

OMPROUTE
Show help on the DISPLAY OMPROUTE command.

OSPF
Show help on the DISPLAY OMPROUTE,OSPF command.

RIP
Show help on the DISPLAY OMPROUTE,RIP command.

RTTABLE
Show help on the DISPLAY OMPROUTE,RTTABLE command.

SYSPLEX
Show help on the DISPLAY SYSPLEX command.

VIPADYN
Show help on the DISPLAY NETSTAT,VIPADYN, and DISPLAY SYSPLEX,VIPADYN commands.

TELNET
Show help on the VARY TELNET command.

ACT
Show help on the VARY TELNET,ACT command.

APPL
Show help on the VARY TELNET,APPL command.

DEFAULTS
Show help on the VARY TELNET,DEFAULTS command.

DEVICETYPE
Show help on the VARY TELNET,DEVICETYPE command.

INACT
Show help on the VARY TELNET,INACT command.

INACTLUS
Show help on the VARY TELNET,INACTLUS command.

IPGROUP
Show help on the VARY TELNET,IPGROUP command.

LUGROUP
Show help on the VARY TELNET,LUGROUP command.

LUMAP
Show help on the VARY TELNET,LUMAP command.

PROFILE
Show help on the VARY TELNET,PROFILE command.

QUIESCE
Show help on the VARY TELNET,QUIESCE command.

RESUME
Show help on the VARY TELNET,RESUME command.

STOP
Show help on the VARY TELNET,STOP command.

WLM
Show help on the VARY TELNET,WLM command.

WHEREUSED
Show help on the VARY TELNET,WHEREUSED command.

Examples

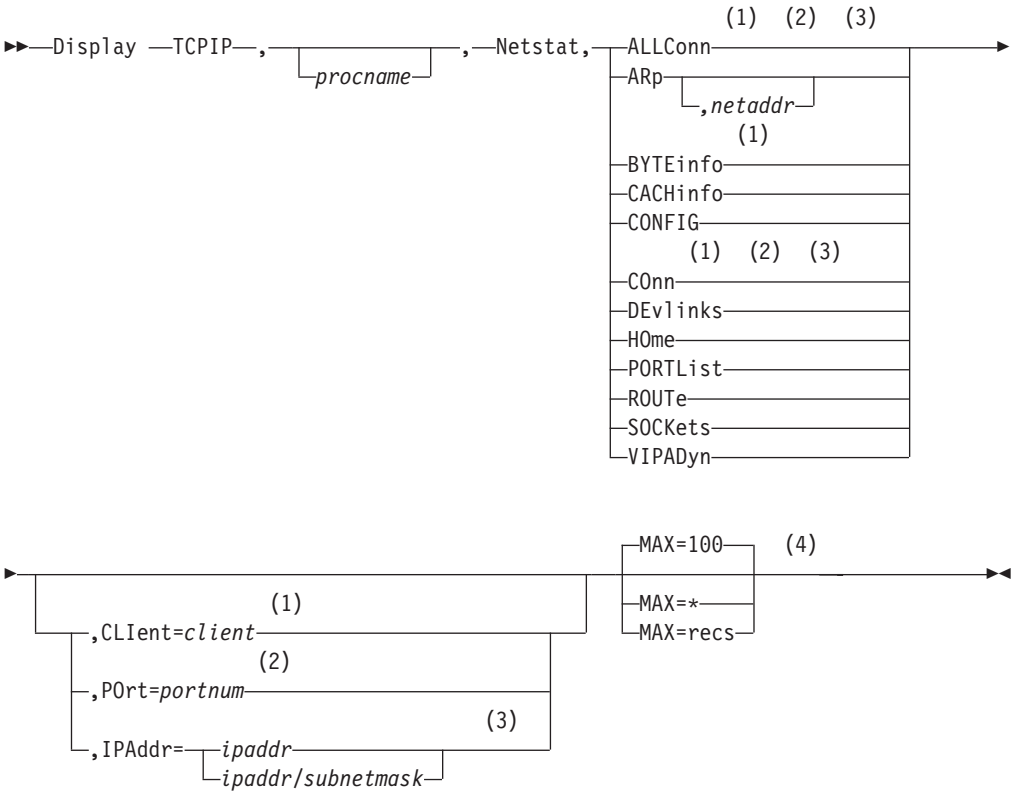
```
d tcpip,tcpa,help,start
EZZ0361I V...(START|CMD=START),XDEVNAME
```

where XDEVNAME is the device name.

DISPLAY TCPIP,,NETSTAT

Use the DISPLAY TCPIP,,NETSTAT command from an operator's console to request NETSTAT information. For a detailed description of each report, see "Using the NETSTAT Command" in the *OS/390 SecureWay Communications Server: IP User's Guide*.

Syntax



Notes:

1 The CLIENT is only valid with ALLCONN, CONN, and BYTEinfo.

2 The PORT filter is only valid on ALLCONN and CONN.

3 The IPADDR is only valid with ALLCONN and CONN.

4 MAX limits the number of records displayed to the MVS operator's console.

Parameters

NETSTAT
Request NETSTAT information.

ALLCONN
Displays information for all TCP/IP connections, including recently closed ones.

ARP
Displays ARP cache information.

netaddr
This field has a maximum length of 15. Format is nnn.nnn.nnn.nnn where nnn is in the range of 0 to 255. You must code all the triplets. No wildcards are allowed.

BYTEINFO
Displays the byte-count information about each connection.

CACHINFO
Displays information about Fast Response Cache Accelerator statistics. Statistics are displayed for each listening socket configured for Fast Response Cache Accelerator support. There will be one section displayed per socket.

CONFIG
Displays TCP/IP configuration data.

CONN
Displays information about each active TCP/IP connection.

DEVLINKS
Displays information about devices and defined links in the TCPIP address space.

HOME
Displays the home list.

PORTLIST
Displays the port reservation list.

ROUTE
Displays routing information in a standard fashion.

SOCKETS
Displays information about each client using the socket interface.

VIPADYN
Displays information about Dynamic VIPA for the active stack. If more than one stack is active, use *procname* to specify the particular TCP stack for which you want to display information.

CLIENT=*client*
Specifies a client ID that is used to limit the ALLCONN, CONN, and BYTEinfo responses. Max size for this field is 8 alphanumeric characters (plus special characters #, $, and @). Wildcards (* and ?) can appear in any position.

IPADDR=*ipaddr*
Specifies an IP address that is used to limit the ALLCONN and CONN options. Max length of this field is 15. Format is nnn.nnn.nnn.nnn, where nnn is in the range 0 to 255. Wildcards (* and ?) are allowed anywhere in the field.

IPADDR=*ipaddr/subnetmask*
Specifies an IP address and subnetmask mask that is used to limit the ALLCONN and CONN options. Format is nnn.nnn.nnn.nnn/nnn.nnn.nnn.nnn, where nnn is in the range 0 to 255. You must code all the triplets. No wildcards are allowed when the subnetmask is specified.

Note: If the subnetmask is not specified, the default of 255.255.255.255 is used.

PORT=*port*
Specifies a port that is used to limit the ALLCONN and CONN responses. The port value range is 0 through 65535. No wildcards are allowed.

MAX=*number of records*

Number of records to be written to the console. Valid range is 1 through 65535. This option is ignored for the CONFIG request since the configuration data is a fixed length.

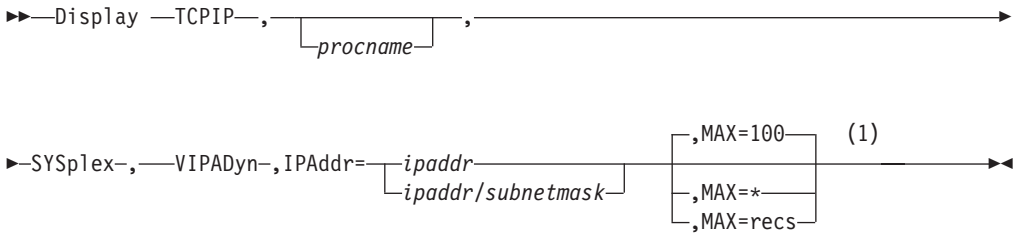
DISPLAY TCPIP,,OMPROUTE

Use the DISPLAY TCPIP,,OMPROUTE command to display OSPF and RIP configuration and state information. For details on the types of data that can be displayed as well as examples of the generated output, see “Using OMPROUTE DISPLAY Commands” on page 993.

DISPLAY TCPIP,,SYSPLEX

Use the DISPLAY TCPIP,,SYSPLEX command from an operator's console to request SYSPLEX information.

Syntax



Notes:

1 MAX limits the number of records displayed to the MVS operator's console.

Parameters

SYSPLEX
Request SYSPLEX information.

VIPADYN
Displays information about Dynamic VIPA for the active stack. If more than one stack is active, use *procname* to specify the particular TCP stack for which you want to display information.

IPADDR=*ipaddr*
Specifies an IP address that is used to limit the VIPADYN option. Max length of this field is 15. Format is nnn.nnn.nnn.nnn, where nnn is in the range 0 to 255. Wildcards (* and ?) are allowed anywhere in the field.

IPADDR=*ipaddr/subnetmask*
Specifies an IP address and subnetmask mask that is used to limit the VIPADYN option. Format is nnn.nnn.nnn.nnn/nnn.nnn.nnn.nnn, where nnn is in the range 0 to 255. You must code all the triplets. No wildcards are allowed when the subnetmask is specified.

Note: If the subnetmask is not specified, the default of 255.255.255.255 is used.

MAX=*number of records*
Number of records to be written to the console. Valid range is 1 through 65535. A wildcard (*) displays all records. Default value is **100**.

Examples

```
D TCPIP,TCPCS,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R8 498
VIPA DYNAMIC DISPLAY FROM TCPCS    AT SYSTEM1
IPADDR: 201.2.10.11  LINKNAME: VIPLC9020A0B
  ORIGIN: VIPADEFINE
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  --------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
```

```
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.12  LINKNAME: VIPLC9020A0C
  ORIGIN: VIPADEFINE
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  -----------      ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  075
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.13
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS2    SYSTEM1   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  100
  TCPCS3    SYSTEM2   BACKUP  010
IPADDR: 201.2.10.21
   ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS2    SYSTEM1   BACKUP  100
  TCPCS     SYSTEM1   BACKUP  080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS3    SYSTEM2   ACTIVE        255.255.255.192  201.2.10.0
  TCPCS     SYSTEM1   BACKUP  080
  TCPCS2    SYSTEM1   BACKUP  075
IPADDR: 201.2.10.193  LINKNAME: VIPLC9020AC1
  ORIGIN: VIPARANGE IOCTL
  TCPNAME   MVSNAME   STATUS  RANK  ADDRESS MASK     NETWORK PREFIX
  --------  --------  ------  ----  ---------------  ---------------
  TCPCS     SYSTEM1   ACTIVE        255.255.255.192  201.2.10.192
16 OF 16 RECORDS DISPLAYED
```

Usage Notes

See to “Chapter 3. Virtual IP Addressing” on page 71 for an explanation of the fields on the report.

DISPLAY TCPIP,,TELNET

Use the DISPLAY TCPIP,,TELNET command from an operator's console to request TELNET information. For further information, see "Operating the Telnet Server Using the DISPLAY Command Set" on page 496.

Using the SMSG Interface

The SMSG interface also allows you to change the characteristics of an active task. This is the general format of SMSG.

Syntax

Parameters

procname
The name of the member in a procedure library that was used to start the server or address space.

parameter
Any of the parameters that are valid for the server.

Usage Notes

The following servers support the MVS SMSG command. Not all servers support the same parameters. You can find further descriptions of the supported parameters in the chapter for that server. Refer to *OS/390 SecureWay Communications Server: IP User's Guide* for information about the SMTP SMG support.

Server/Addr Space	Supported Parameters
SMTP	DEBUG, EXPIRE, HELP, NODEBUG, NOTRACE, QUEUES, SHUTDOWN, STATS, TRACE
Remote Print Server (LPD)	PRINT WORK, TRACE OFF, TRACE ON

Chapter 6. Defining the TCP/IP Client System Parameters

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how you can define the TCP/IP system parameters required by the client programs. These parameters are specified using the configuration statements in a TCPIP.DATA file.

Configuration Process

You must configure a file containing TCPIP.DATA statements. The OS/390 UNIX search path for a file that contains these statements is:

- Environment variable RESOLVER_CONFIG
- The HFS file /etc/resolv.conf
- Non-OS/390 UNIX search path

Note: In most cases, the OS/390 UNIX search path does not include jobname.TCPIP.DATA, but only searches for userid. In the case of TSO and in some cases, OMVS, the jobname is the userid.

The non-OS/390 UNIX search path varies depending on the execution environment of the application needing access to a TCPIP.DATA file. For help in setting up multiple copies of TCPIP.DATA, see “Considerations for Multiple Instances of TCP/IP” on page 57. A single TCP/IP copy (or instance) needs a TCPIP.DATA file for the instance and all local applications that use the file to be configured correctly.

Create a TCPIP.DATA file by copying the sample provided in TCPIP.SEZAINST(TCPDATA) and modifying it to suit your local conditions.

Allocate this data set with either sequential (PS) or partitioned (PO) organization, a fixed block format (FB), a logical record length (LRECL) of 80, and any valid blocksize value for a fixed block, such as 3120. This file can also be the HFS file /etc/resolv.conf, or an HFS file that is pointed to by either the environment variable RESOLVER_CONFIG or the SYSTCPD DD in a JCL procedure. The environment variable RESOLVER_CONFIG can also point to an MVS data set or PDS.

You can use any name for the TCPIP.DATA data set if you access it using the //SYSTCPD DD statement, or use ENVAR to set RESOLVER_CONFIG,in the JCL for all the servers, logon procedures, and batch jobs that execute TCP/IP functions. If you are not using the //SYSTCPD DD statement, the environment variable, or /etc/resolv.conf,then the data set name must conform to the conventions described in “Information Specific to Data Sets in Configuration File Search Orders” on page 14. Another alternative is to use the well-known data set name SYS1.TCPPARMS(TCPDATA). You can issue the HOMETEST command to verify the actual name of the data set name the system finds for TCPIP.DATA.

Note: If either the environment variable RESOLVER_CONFIG or /etc/resolv.conf is being used or jobname is not the same as userid and the OS/390 resolver

obtains the information from the userid, then HOMETEST is of no use because OS/390 UNIX extensions to the traditional TCP/IP search path will not be understood.

Each configuration statement can be preceded by an optional *system_name*. This permits configuration information for multiple systems to be specified in a single *hlq*.TCPIP.DATA data set. The *system_name* is matched against the name of the system on which you are running. The name of the system is taken from the node name in the IEFSSNxx member of PARMLIB.

The statements are processed in the order they appear in the data set. The following rules apply to this processing.

- If the *system_name* does not match the name of the system, the configuration statement is ignored.
- If *system_name* is blank, the configuration statement is in effect on every system.
- If the *system_name* matches the node name, the configuration statement that follows it is in effect.
- The ***last*** statement that matches is effective.

For example, if you have the following three TCPIPJOBNAME statements, MVS6 would look for a TCPIP cataloged procedure named TCPBTA2, MVSA would look for TCPV3, and all other systems would look for TCPMCWN.

```
      TCPIPJOBNAME TCPMCWN
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
```

But if you reversed the order, all systems would try to find the procedure named TCPMCWN.

```
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
      TCPIPJOBNAME TCPMCWN
```

Summary of Statements in TCPIP.DATA

The statements and system parameters are summarized in Table 13.

Notes:

1. The LE resolver supports the DOMAIN statement, which is treated the same as the DOMAINORIGIN statement, and the NAMESERVER statement, which is treated the same as the NSINTERADDR statement. The TCP/IP resolver does not process these statements. Use of the DOMAIN and NAMESERVER statements in configuration files processed by both the LE resolver and the TCP/IP resolver can lead to inconsistent behavior between applications, based on which resolver is used by the applications.
2. The LE resolver uses two keywords, DOMAIN and NAMESERVER, which are not supported by the resolver (TCP/IP resolver). This discrepancy can lead to inconsistent behavior among applications if they use different resolvers.

Table 13. Summary of TCPIP.DATA Configuration Statements

Statement	Description	Page
ALWAYSWTO	Issue WTO messages for all servers	292
DATASETPREFIX	Set the high-level qualifier for dynamic allocation of data sets	293
DOMAINORIGIN	Specify the domain origin that is appended to the host name to form the fully qualified domain name for a host	294

Table 13. Summary of TCPIP.DATA Configuration Statements (continued)

Statement	Description	Page
HOSTNAME	Specify the TCP host name of the OS/390 server	295
LOADDBCSTABLES	Tell FTP which DBCS translation tables can be loaded	296
MESSAGECASE	Specify case translation for the FTP server and osnmpd	298
NSINTERADDR	Define the IP address of a name server in dotted decimal format	299
NSPORTADDR	Specify the name server port number	300
RESOLVEVIA	Specify the protocol used by the resolver to communicate with the name server	301
RESOLVERTIMEOUT	Specify how long the resolver waits for a response while trying to communicate with the name server	302
RESOLVERUDPRETRIES	Specify how many times the resolver tries to connect to the name server with when using UDP datagrams	303
SOCKDEBUG	Turn on tracing of socket library calls	304
SOCKNOTESTSTOR	Stop checking of socket calls for storage access errors on the parameters to the call	305
TCPIPJOBNAME	Specify the member name of the cataloged procedure used to start the TCPIP address space	306
TRACE RESOLVER	Trace all queries to and responses from the name server	307
TRACE SOCKET	Trace C socket library calls to TCP/IP	308

Sample TCPIP.DATA Data Set (TCPDATA)

The following sample is used to specify configuration information of client parameters.

```
;*********************************************************************
;                                                                    *
;   Name of Data Set:     TCPIP.DATA                                 *
;                                                                    *
;   COPYRIGHT = NONE.                                                *
;                                                                    *
;   This data, TCPIP.DATA, is used to specify configuration          *
;   information required by TCP/IP client and server programs.       *
;                                                                    *
;                                                                    *
;   Syntax Rules for the TCPIP.DATA configuration data set:          *
;                                                                    *
;   (a) All characters to the right of and including a ; will be     *
;       treated as a comment.                                        *
;                                                                    *
;   (b) Blanks and  are used to delimit tokens.                      *
;                                                                    *
;   (c) The format for each configuration statement is:              *
;                                                                    *
;         keyword  value                                             *
;                                                                    *
;       where  is an optional label that can be                      *
;       specified before a keyword; if present, then the keyword-    *
;       value pair will only be recognized if the SystemName matches *
;       the node name of the system, as defined in the IEFSSNxx      *
;       PARMLIB member.  This optional label permits configuration   *
;       information for multiple systems to be specified in a single *
;       TCPIP.DATA data set.                                         *
;                                                                    *
;       NOTE:  You should define the SystemName in the IEFSSNxx      *
;              PARMLIB member to be the same as your JES2 or JES3    *
```

```
;              node name.  This is required for correct delivery of    *
;              SMTP mail.                                               *
;                                                                       *
;   (d) If a statement is specified multiple times, the last statement *
;       is effective.                                                   *
;                                                                       *
;************************************************************************
;
; TCPIPJOBNAME statement
; ======================
; TCPIPJOBNAME specifies the name of the started procedure that was
; used to start the TCPIP address space.    TCPIP is the default.
;
; If multiple TCPIP stacks are run on a single system, each stack will
; require its own copy of this file, each with a different value for
; TCPIPJOBNAME.
;
TCPIPJOBNAME TCPIP
;
;
; HOSTNAME statement
; ==================
; HOSTNAME specifies the TCP host name of this system as it is known
; in the IP network.  If not specified, the default HOSTNAME will be
; the node name specified in the IEFSSNxx PARMLIB member.
;
; For example, if this TCPIP.DATA data set is shared between 2
; systems, OURMVSNAME and YOURMVSNAME, then the following 2 lines
; will define the HOSTNAME correctly on each system.
;
; OURMVSNAME:    HOSTNAME  OURTCPNAME
; YOURMVSNAME:   HOSTNAME  YOURTCPNAME
;
; No prefix is required if the TCPIP.DATA file is not being shared.
;
HOSTNAME THISTCPNAME
;
; DOMAINORIGIN statement
; ======================
; DOMAINORIGIN specifies the domain origin that will be appended
; to host names passed to the resolver.  If a host name contains
; any dots, then the DOMAINORIGIN will not be appended to the
; host name.
;
DOMAINORIGIN  YOUR.DOMAIN.NAME
;
;
; DATASETPREFIX statement
; =======================
; DATASETPREFIX is used to set the high level qualifier for dynamic
; allocation of datasets in TCP/IP.
;
; The character string specified as a parameter on
; DATASETPREFIX takes precedence over the default prefix of "TCPIP".
;
; The DATASETPREFIX parameter can be up to 26 characters long
; and the parameter must NOT end with a period.
;
; For more information please see "Dynamic Data Set Allocation" in
; the IP Configuration Guide.
;
DATASETPREFIX TCPIP
;
;
; MESSAGECASE statement
; =====================
; MESSAGECASE MIXED indicates to some servers, such as FTPD, that
```

```
; messages should be displayed in mixed case.  MESSAGECASE UPPER
; indicates that all messages should be displayed in uppercase.  Mixed
; case strings that are inserted in messages will not be uppercased.
;
; If MESSAGECASE is not specified, mixed case messages will be used.
;
; MESSAGECASE MIXED
; MESSAGECASE UPPER
;
;
; NSINTERADDR statement
; ====================
; NSINTERADDR specifies the IP address of the name server.
; LOOPBACK (127.0.0.1) specifies your local name server.  If a name
; server will not be used, then do not code an NSINTERADDR statement.
; (Comment out the NSINTERADDR line below).  This will cause all names
; to be resolved via site table lookup.
;
; The NSINTERADDR statement can be repeated as many times as you need
; to specify alternate name servers.  The name server listed first
; will be the first one attempted.
;
; NSINTERADDR  127.0.0.1
;
;
; NSPORTADDR statement
; ====================
; NSPORTADDR specifies the foreign port of the name server.
; 53 is the default value.
;
; NSPORTADDR 53
;
;
; RESOLVEVIA statement
; ====================
;
; RESOLVEVIA specifies how the resolver is to communicate with the
; name server.  TCP indicates use of TCP connections.  UDP indicates
; use of UDP datagrams.  The default is UDP.
;
RESOLVEVIA UDP
;
;
; RESOLVERTIMEOUT statement
; ========================
; RESOLVERTIMEOUT specifies the time in seconds that the resolver
; will wait for a response from the name server (either UDP or TCP).
; The default is 30 seconds.
;
RESOLVERTIMEOUT 10
;
;
; RESOLVERUDPRETRIES statement
; ===========================
;
; RESOLVERUDPRETRIES specifies the number of times the resolver
; should try to connect to the name server when using UDP datagrams.
; The default is 1.
;
RESOLVERUDPRETRIES 1
;
;
; LOADDBCSTABLES statement
; =======================
; LOADDBCSTABLES indicates to the FTP server and FTP client which DBCS
; translation tables should be loaded at initialization time. Remove
; from the list any tables that are not required. If LOADDBCSTABLES is
```

```
; not specified, no DBCS tables will be loaded.
;
; LOADDBCSTABLES JIS78KJ JIS83KJ SJISKANJI EUCKANJI HANGEUL KSC5601
; LOADDBCSTABLES TCHINESE BIG5 SCHINESE
;
;
; SOCKDEBUG statement
; ===================
; SOCKDEBUG will cause a trace of socket library calls to be written.
; This command is for debugging purposes only.
;
; SOCKDEBUG
;
;
; SOCKNOTESTSTOR statement
; ========================
; SOCKTESTSTOR is used to check socket calls for storage access errors
; on the parameters to the call.  SOCKNOTESTSTOR stops this checking
; and is better for response time.  SOCKNOTESTSTOR is the default.
;
; SOCKTESTSTOR
; SOCKNOTESTSTOR
;
;
; TRACE RESOLVER statement
; ========================
; TRACE RESOLVER will cause a complete trace of all queries to and
; responses from the name server or site tables to be written to
; the user's console.  This command is for debugging purposes only.
;
; TRACE RESOLVER
;
;
; TRACE SOCKET statement
; ======================
; TRACE SOCKET will cause a complete trace of all calls to TCP/IP
; though the C socket library to the SYSPRINT DD dataset.
; This statement is for debugging purposes only.
;
; TRACE SOCKET
;
;
; ALWAYSWTO statement
; ===================
; ALWAYSWTO causes messages for some servers, such as SMTP and LPD,
; to be issued as WTOs.
;
; ALWAYSWTO YES
;
; Obsolete statements
; ===================
; The following statements no longer have any effect when included in
; this file:
;   SOCKBULKMODE
;   SOCKDEBUGBULKPERF0
;
; End of file.
;
```

TCPIP.DATA Configuration Statements

This section explains each statement for the TCPIP.DATA data set in detail.

Syntax Conventions

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

ALWAYSWTO Statement

For some servers, such as SMTP and LPD, use the ALWAYSWTO statement to have TCP/IP issue WTO messages for some servers.

Syntax

Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,nodename,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

YES
Indicates that all server messages are to be displayed on the console.

Examples

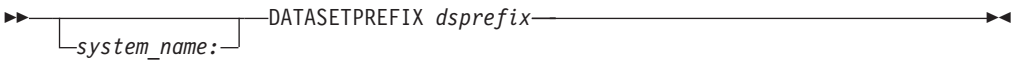
Have TCP/IP send the WTO messages on the MVSMFG2 system.

```
MVSMFG2:ALWAYSWTO YES
```

DATASETPREFIX Statement

Use the DATASETPREFIX statement to set the high-level qualifier for the dynamic allocation of data sets in TCP/IP.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

dsprefix
The prefix to use as the high-level qualifier for the dynamic allocation of data sets. The default high-level qualifier distributed with the system is TCPIP.

Examples

Set the data set prefix for all client and server data sets:

```
DATASETPREFIX TCPIP.V2R7
```

Usage Notes

The DATASETPREFIX in TCPIP.DATA is used by all clients and all servers except the TCPIP address space.

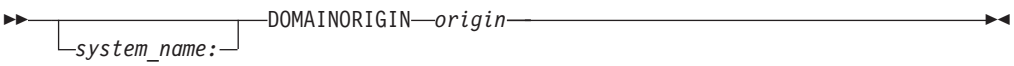
Related Topics

"Configuration Data Sets and HFS Files" on page 13

DOMAINORIGIN Statement

Use the DOMAINORIGIN statement to specify the domain origin that is appended to the host name to form the fully qualified domain name for a host.

Syntax



Parameters

system_name:
: The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

origin
: The domain origin is appended to the host name. This name usually has imbedded dots.

Examples

This example appends the domain origin of BOBS.YOUR.UNCLE to the host name:

```
DOMAINORIGIN  BOBS.YOUR.UNCLE
```

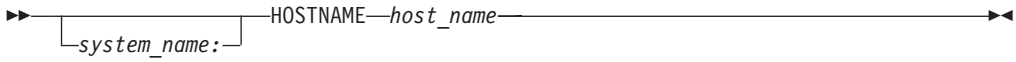
Usage Notes

- No case translation is performed on the domain origin.
- If the resolver is passed a host name that does not contain any dots (in dotted decimal notation), the domain origin is appended to the host name. If the host name passed to the resolver contains dots, the domain origin is not appended to the host name.
- The DOMAINORIGIN configuration statement must be customized at each site.

HOSTNAME Statement

Use the HOSTNAME statement to specify the TCP host name of this OS/390 server. 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).

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

host_name
The host name. If not specified, defaults to the *nodename* specified in the IEFSSNxx PARMLIB member.

Examples

The TCPIP.DATA data set will be shared between 2 systems, MVSMFG4 and MVSADM1. The HOSTNAME statements define the host name on each system.

```
MVSMFG4: HOSTNAME MVSMFG4
MVSADM1: HOSTNAME MVSADM1
```

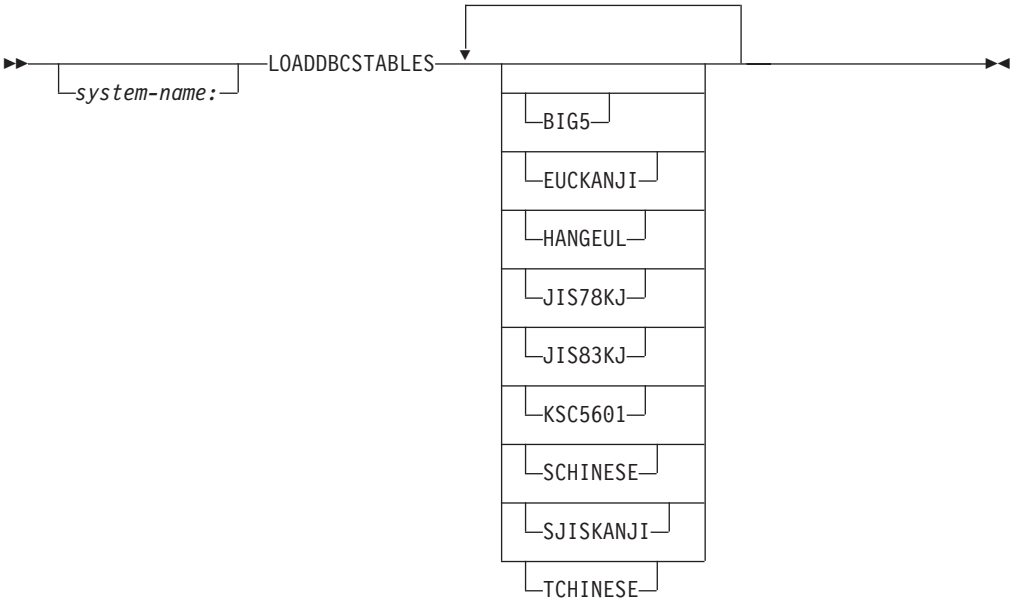
Usage Notes

Case translation is not performed on the host name.

LOADDBCSTABLES Statement

Use the LOADDBCSTABLES statement to tell the FTP server and client which DBCS translation tables can be loaded.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

BIG5
Indicates to the FTP server and client that the BIG5 DBCS translation table should be loaded from the TCPCHBIN binary translate table data set.

EUCKANJI
Indicates to the FTP server and client that the Extended Unix Code Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

HANGEUL
Indicates to the FTP server and client that the Hangeul DBCS translation table should be loaded from the TCPHGBIN binary translate table data set.

JIS78KJ
Indicates to the FTP server and client that the JIS 1978 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

JIS83KJ
Indicates to the FTP server and client that the JIS 1983 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

KSC5601
Indicates to the FTP server and client that the Korean Standard Code KSC-5601 DBCS translation table should be loaded from the TCPHGBIN binary translate table data set.

SCHINESE
Indicates to the FTP server and client that the Simplified Chinese DBCS translation table should be loaded from the TCPSCBIN binary translate table data set.

SJISKANJI
Indicates to the FTP server and client that the Shift JIS Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

TCHINESE
Indicates to the FTP server and client that the Traditional Chinese (5550) DBCS translation table should be loaded from the TCPCHBIN binary translate table data set.

Examples

Load the Korean Standard Code KSC-5601 and the Traditional Chinese (5550) DBCS translation tables:

```
LOADDBCSTABLES KSC5601 TCHINESE
```

Usage Notes

- You can select any or all the of translation tables or specify none. However, additional virtual storage may be required by the FTP server and client when a large number of translation tables are loaded at the same time.
- All the parameters must fit one line. You can repeat the LOADDBCSTABLES statement as necessary to specify additional tables to be loaded.
- If the LOADDBCSTABLES parameter is not specified, is specified incorrectly, or if *hlq*.TCPIP.DATA is not accessible, then no DBCS translation tables will be loaded, and the corresponding FTP server and client DBCS transfer types will be unavailable.
- The IBMKANJI transfer type does not require any translation table to be loaded.

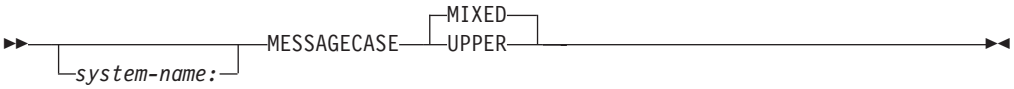
Related Topics

“Using Translation Tables” in the OS/390 SecureWay Communications Server: IP Configuration.

MESSAGECASE Statement

Use the MESSAGECASE statement to specify whether to convert Write To Operator (WTO) messages into uppercase for the FTP server and the osnmpd server.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

MIXED
Indicates to the FTP server and the osnmpd server that all WTO messages should be displayed in mixed case.

UPPER
Indicates to the FTP server and the osnmpd server that all WTO messages should be displayed in uppercase.

Examples

Display all messages to the MVSTEST system in upper case:

```
MVSTEST: MESSAGECASE UPPER
```

Usage Notes

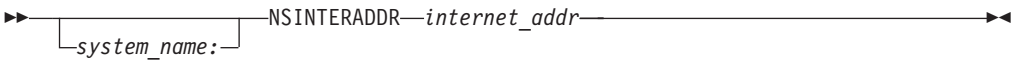
- If you specify MIXED, no case conversion is performed on WTO messages.
- If the MESSAGECASE statement is not specified, is specified incorrectly, if MIXED or UPPER are not specified, or if *hlq*.TCPIP.DATA is not accessible, then mixed case messages will be displayed.
- Any WTO messages that are displayed by the FTP server and the osnmpd agent at initialization, ***before*** *hlq*.TCPIP.DATA is read, will be displayed in uppercase.
- All WTO messages issued by the TCPIP stack will be displayed in upper case and are not affected by the MESSAGECASE value.
- Additionally, the MESSAGECASE statement can be set from the OS/390 shell environment by exporting the MESSAGECASE environment variable.

The setting of the MESSAGECASE environment variable overrides any setting found in TCPIP.DATA. If MESSAGECASE is not defined as a environment variable or as a statement in TCPIP.DATA, the WTO message will remain in mixed case.

NSINTERADDR Statement

Use the NSINTERADDR statement to define the IP address of a name server in dotted decimal format.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

internet_addr
The IP address of a name server.

Examples

Specify the IP address of the name server to be 14.13.12.11:

```
NSINTERADDR  14.13.12.11
```

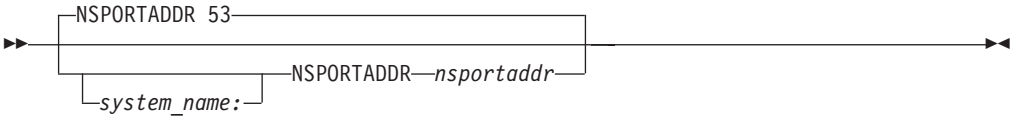
Usage Notes

- You can repeat this statement as many times as you need to specify the IP addresses of alternative name servers.
- Connections to the name servers are attempted in the order they appear in the *hlq*.TCPIP.DATA data set.
- If no NSINTERADDR statements are coded in the *hlq*.TCPIP.DATA data set, the resolver looks for all domain names in the site table, and does not attempt to use a name server.

NSPORTADDR Statement

Use the NSPORTADDR statement to specify the name server port number.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

nsportaddr
The name server port number. The default is port 53.

Examples

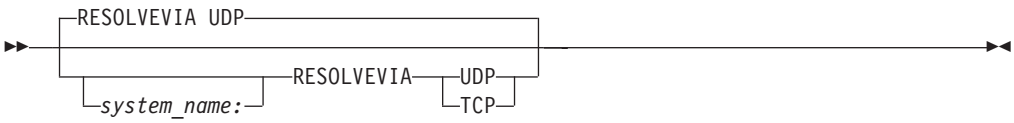
Specify the foreign port of the name server to be 55:

```
NSPORTADDR 55
```

RESOLVEVIA Statement

Use the RESOLVEVIA statement to specify the protocol used by the resolver to communicate with the name server.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

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

TCP
Specifies that the protocol is TCP.

Examples

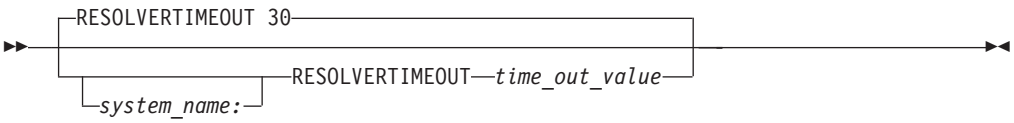
Specify that the resolver is to communicate with the name server using TCP virtual circuits:

```
RESOLVEVIA TCP
```

RESOLVERTIMEOUT Statement

Use the RESOLVERTIMEOUT statement to specify the number of seconds the resolver waits for a response while trying to communicate with the name server (either UDP or TCP).

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

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

Examples

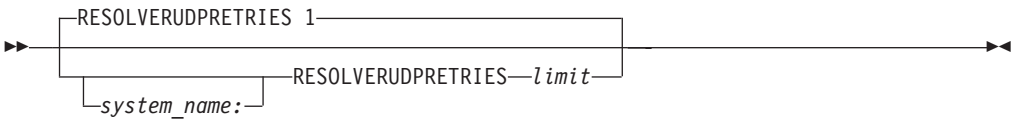
Specify a 10 second waiting time for the resolver when completing an open to the name server:

```
RESOLVERTIMEOUT 10
```

RESOLVERUDPRETRIES Statement

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

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

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

Examples

Specify 2 as the number of times the resolver will try to connect to the name server when using UDP datagrams:

```
RESOLVERUDPRETRIES 2
```

SOCKDEBUG Statement

Use the SOCKDEBUG statement to turn on the tracing of socket library calls. This statement only produces trace message for sockets using the TCP/IP C sockets or TCP/IP REXX sockets application programming interfaces.

Syntax

Parameters

None.

Usage Notes

This statement works for all sockets across the system the way sock_debug() works for a specific socket application.

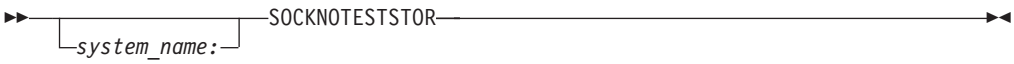
Related Topics

See the *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide* for more information on sockets.

SOCKNOTESTSTOR Statement

Use the SOCKNOTESTSTOR statement to stop checking of socket calls for storage access errors on the parameters to the call.

Syntax



Parameters

None.

Usage Notes

- This statement will improve response time.
- This statement works for all sockets across the system the way sock_do_test_stor() works for a specific socket application.

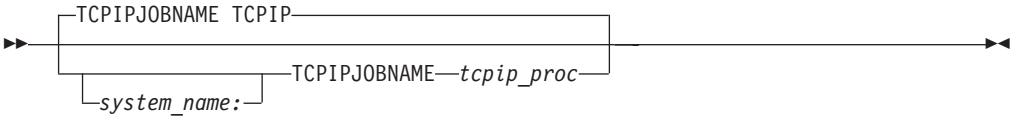
Related Topics

- See the *OS/390 SecureWay Communications Server: IP Application Programming Interface Guide* for more information on sockets.

TCPIPJOBNAME Statement

Use the TCPIPJOBNAME statement to specify the member name of the procedure used to start the TCPIP address space.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

tcpip_proc
The name of the member in the cataloged procedure library that is used to start the TCPIP address space. The default is TCPIP.

Examples

Specify TCPIPA as the name of the procedure that was used to start the TCPIP address space:

```
TCPIPJOBNAME TCPIPA
```

Usage Notes

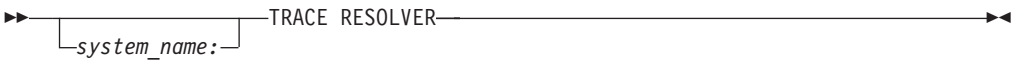
You must specify the proper data set name of the TCPIP address space on your system (for example, c/data set name/procedure name/). If *tcpip_proc* is not the name of the started TCPIP address space, clients will fail at startup with an irrecoverable interaddress communication error.

For more information about why the TCPIPJOBNAME parameter must match the name of the associated TCP/IP address space and be the same name as that defined for the corresponding AF_INET physical file system in the BPXPRMxx member used to configure OS/390 UNIX, see “Considerations for Multiple Instances of TCP/IP” on page 57.

TRACE RESOLVER Statement

Use the TRACE RESOLVER statement to have a complete trace of all queries to and responses from the name server to be written to the user's console.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*nodename*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

Examples

Do a complete trace of all queries to and from the name server:

```
TRACE RESOLVER
```

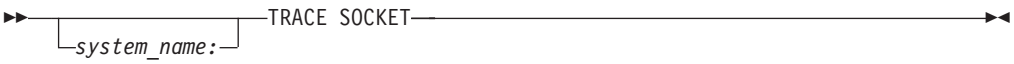
Usage Notes

The TRACE RESOLVER statement is used for debugging purposes only.

TRACE SOCKET Statement

Use the TRACE SOCKET statement to have a complete trace of all calls to TCP/IP through the C socket library.

Syntax



Parameters

system_name:
The system name is derived from the line containing the definition, `VMCF,MVPXSSI,`*`nodename`*`,` in the IEFSSNxx member of PARMLIB. This parameter should be set to the same name as your JES NJE *nodename*. The colon is required.

Examples

Do a complete trace of all C socket calls to TCP/IP:

```
TRACE SOCKET
```

Usage Notes

The TRACE SOCKET statement is used for debugging purposes only.

The output from the TRACE SOCKET command is sent to the data set referred to by the SYSPRINT DD statement.

Chapter 7. Configuring the Site Table

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

Also, certain socket calls related to name and address resolution are affected by the OS/390 UNIX search path and resolution process. For more information, see *OS/390 UNIX System Services Planning*. Also see the *OS/390 SecureWay Communications Server: IP Migration*.

The site table is generated from the *hlq*.HOSTS.LOCAL data set. This data set contains descriptions of local host entries in the HOSTS format. A sample HOSTS.LOCAL data set is created during installation.

This chapter describes how to update the sample *hlq*.HOSTS.LOCAL data set and use it to generate the two data sets, *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, which function as your site table. It also explains how to test the site table when you are done.

To insure that the TCPIP.DATA statements that affect the resolution services are properly defined for your configuration, see “Chapter 6. Defining the TCP/IP Client System Parameters” on page 285.

The OS/390 UNIX search path for HOSTS.SITEINFO configuration is:

- Value of the environment variable X_SITE (if set).

  This environment variable can contain the information that, in a non-OS/390 UNIX environment, was contained in a HOSTS.SITEINFO file that was created by MAKESITE.
- The HFS file /etc/hosts (if it exists).

  This is the equivalent, default, OS/390 UNIX file that can be used in place of a HOSTS.SITEINFO file.
- *userid*.HOSTS.SITEINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
- *tcpip*.HOSTS.SITEINFO

  tcpip represents the value of the DATASETPREFIX statement in a TCPIP.DATA file. The default is TCPIP.

HOSTS.SITEINFO information is used by the following functions:

- gethostbyname()
- sethostent()
- gethostent()
- endhostent()
- getnetbyname()

The OS/390 UNIX search path for HOSTS.ADDRINFO configuration is:

- Value of the environment variable X_ADDR (if set).

  This environment variable can contain the information that, in a non-OS/390 UNIX environment, was contained in a HOSTS.ADDRINFO file that was created by MAKESITE.

- The HFS file /etc/hosts (if it exists).

  This is the equivalent, default OS/390 UNIX file that can be used in place of a HOSTS.ADDRINFO file.
- *userid*.HOSTS.ADDRINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
- *tcpip*.HOSTS.ADDRINFO

  tcpip represents the value of the DATASETPREFIX statement in a TCPIP.DATA file. The default is TCPIP.

HOSTS.ADDRINFO information is used by the following functions:

- getnetbyaddr()
- setnetent()
- getnetent()
- endnetent()
- gethostbyaddr()

Note that if /etc/hosts exists, it overrides both a HOSTS.SITEINFO data set and a HOSTS.ADDRINFO data set since it contains both types of configuration.

Configuration Process

Note: If you are not using the OS/390 UNIX environment variables X_SITE and X_ADDR, or the HFS file /etc/hosts, use the following steps to configure your site table:

1. Update the HOSTS.LOCAL data set
2. Run MAKESITE

Step 1: Update the HOSTS.LOCAL Data Set

A sample HOSTS.LOCAL data set is distributed with TCP/IP for MVS. The installation job, EZAGETIN, copies this sample data set from *hlq*.SEZAINST(HOSTS) to *hlq*.HOSTS.LOCAL for you. Because each site is unique and requires customized statements, you should only use this data set as a guideline. Update the HOST, NET, and GATEWAY entries in this data set to suit your installation.

HOST Entries

One line of the *hlq*.HOSTS.LOCAL data set is used for each distinct host and ends with 4 colons. Each host can have multiple IP addresses and multiple names. The line for each host has 3 essential fields, separated by colons. These fields are:

- The keyword *HOST*
- A list, separated by commas, of IP addresses for that host
- A list, separated by commas, of fully qualified names for that host

For example, if you have 2 local hosts, LOCAL1 (IP addresses 192.6.77.4 and 192.8.4.1) and LOCAL2 (with an alias LOCALB and IP address 192.6.77.2), append the following lines to the *hlq*.HOSTS.LOCAL data set:

```
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. However, the name server does not have a maximum character length.

NET and GATEWAY Entries

The NET and GATEWAY statements are not used by TCP/IP for MVS applications. However, some socket calls require the NET entries. If your programs do not need the NET and GATEWAY statements, delete them before invoking MAKESITE.

Sample HOSTS.LOCAL Data Set (HOSTS)

Following is the sample HOSTS.LOCAL data set:

```
; HOSTS.LOCAL
; -----------
; COPYRIGHT = NONE.
;
; The format of this file is documented in RFC 952, "DoD Internet
; Host Table Specification".
;
; The format for entries is:
;
; NET : ADDR : NETNAME :
; GATEWAY : ADDR, ALT-ADDR : HOSTNAME : CPUTYPE : OPSYS : PROTOCOLS :
; HOST : ADDR, ALT-ADDR : HOSTNAME, NICKNAME : CPUTYPE : OPSYS : PROTOCOLS :
;
; Where:
;   ADDR, ALT-ADDR = IP address in decimal, e.g., 26.0.0.73
;   HOSTNAME, NICKNAME = the fully qualified host name and any nicknames
;   CPUTYPE = machine type (PDP-11/70, VAX-11/780, IBM-3090, C/30, etc.)
;   OPSYS = operating system (UNIX, TOPS20, TENEX, VM/SP, etc.)
;   PROTOCOLS = transport/service (TCP/TELNET,TCP/FTP, etc.)
;   : (colon) = field delimiter
;   :: (2 colons) = null field
; *** CPUTYPE, OPSYS, and PROTOCOLS are optional fields.
;
;   MAKESITE does not allow continuation lines, as described in
;   note 2 of the section "GRAMMATICAL HOST TABLE SPECIFICATION"
;   in RFC 952.  Entries should be specified on a single line of
;   up to a maximum of 512 characters per line.
;
;
; Note:  The NET and GATEWAY statements are not used by the TCP/IP for
;        MVS applications.  However, some socket calls require the NET
;        entries.  For better performance, if your programs do not need
;        the NET and GATEWAY statements, delete them before running
;        the MAKESITE program.
;
;
HOST : 9.67.43.100 : NAMESERVER ::::
HOST : 9.67.43.126 : RALEIGH ::::
HOST : 129.34.128.245, 129.34.128.246 : YORKTOWN, WATSON ::::
;
NET  : 9.67.43.0 : RALEIGH.IBM.COM :
;
GATEWAY : 129.34.0.0 : YORKTOWN-GATEWAY ::::
;
```

Step 2: Run MAKESITE

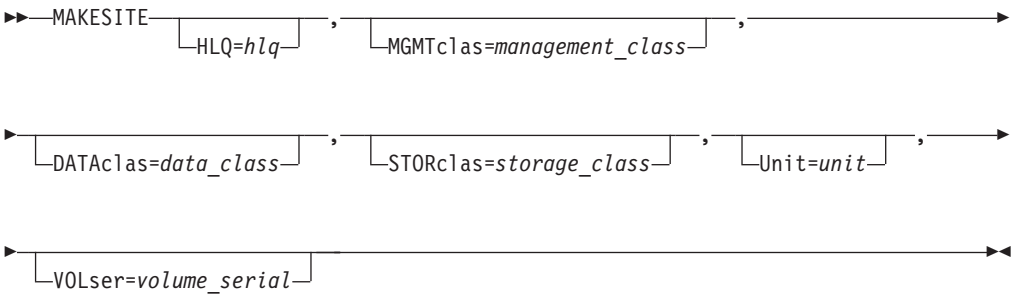
After you make changes to your *hlq*.HOSTS.LOCAL data set, you must generate and install new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets.

Because many servers and commands allocate *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, it is important not to overwrite or delete these data sets while TCP/IP is running. To avoid disrupting any active users, use an HLQ that is different than your active HLQ. This will allow you to swap names (by renaming the old HOSTS data sets and then renaming the new HOSTS data sets) without starting and stopping TCP/IP.

MAKESITE Command

Use MAKESITE as a TSO command or in a batch job to generate new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets. The parameters are the same for either a TSO command or a batch job invocation of MAKESITE.

Syntax



Parameters

HLQ=*hlq*

The high-level qualifier of both the input and output data sets. The name specified is appended to the HOSTS.LOCAL, HOSTS.SITEINFO and HOSTS.ADDRINFO data set names.

Minimum abbreviation: HLQ=,
Maximum length: 29 characters

MGMTCLAS=*management_class*

The SMS-managed management class. MGMTCLAS is valid only in an SMS environment.

Minimum abbreviation: MGMT=
Maximum length: 8 characters

DATACLAS=*data_class*

The SMS-managed data class. DATACLAS is valid only in an SMS environment.

Minimum abbreviation: DATA=
Maximum length: 8 characters

STORCLAS=*storage_class*

The SMS-managed storage class. STORCLAS is valid only in an SMS environment.

Minimum abbreviation: STOR=
Maximum length: 8 characters

UNIT=*unit*

An esoteric device name.

Minimum abbreviation: U=
Maximum length: 8 characters

VOLSER=*volume_serial*
Volume serial number.

Minimum abbreviation: VOL=
Maximum length: 6 characters

Usage Notes

- The optional parameters can be in any order
- Blanks are not allowed in the syntax
- MAKESITE gets its input from *hlq*.HOSTS.LOCAL, where the HLQ is derived in this order:
  - HLQ parameter specified either with the command or in the batch job
  - TSO userid or the TSO PROFILE PREFIX, if it is different from the userid. In a batch job, *userid* can come from any of several sources depending on the environment. It can be the user ID of the user who submitted the batch job, or it can be the batch job name.
  - The value specified with the DATASETPREFIX statement in TCPIP.DATA
  - System default

  The output data sets produced by MAKESITE are prefixed by either the HLQ parameter specified either on the command or batch job or the TSO userid or TSO PROFILE PREFIX, if it is different from the userid.
- If any MAKESITE parameters are specified incorrectly, MAKESITE still executes using defaults (for example, for an incorrect *hlq*, the default is the active userid or jobname).
- Components that use the output from MAKESITE follow the standard naming conventions. If a DATASETPREFIX has been specified, it will be used as the high-level qualifier for HOSTS.SITEINFO and HOSTS.ADDRINFO.

Examples

If your current active HLQ was TCPIP.MVSA, you would follow these steps to run MAKESITE and rename the output data sets.

1. Run MAKESITE with the appropriate parameters to generate 2 new data sets from the new *hlq*.HOSTS.LOCAL data set.

   As a TSO command, you might enter:

   ```
   MAKESITE HLQ=TCPIP.H0004,MGMT=M0001,VOLSER=STRG01,UNIT=SYSDA
   ```

   As a batch job, you might use this JCL:

   ```
   //MAKESITE JOB ,TIME=2,NOTIFY=USER7
   //*
   //BATCH  EXEC PGM=MAKESITE,REGION=8000K,
   //  PARM='VOLSER=STRG01,UNIT=SYSDA,HLQ=TCPIP.H0004,MGMT=M0001'
   //*
   //STEPLIB DD DISP=SHR,DSN=TCPIP.SEZALINK
   //SYSPRINT  DD  SYSOUT=*,DCB=(LRECL=132,RECFM=FBA,BLKSIZE=3960)
   //SYSABEND  DD  SYSOUT=*
   //
   ```

   Note the following:

   - This JCL is not shipped with TCP/IP
   - The size of the parameter string is limited to 100 bytes
   - Keywords in the parameter string can be abbreviated as shown in the MAKESITE syntax descriptions

- Region size varies according to your configuration. Make sure that the region size specified is valid for your configuration.

This will create TCPIP.H004.HOSTS.SITEINFO and TCPIP.H0004.HOSTS.ADDRINFO.

2. Rename your existing HOSTS.SITEINFO and HOSTS.ADDRINFO data sets. These data sets are currently accessed by TCP/IP users on the system and should not be deleted while TCP/IP is running.

   For example, change TCPIP.MVSA.HOSTS.SITEINFO to TCPIP.MVSA.HOSTS.SITEOLD and TCPIP.MVSA.HOSTS.ADDRINFO to TCPIP.MVSA.HOSTS.ADDROLD.

3. Rename the new HOSTS.ADDRINFO and HOSTS.SITEINFO data sets to replace the old ones.

   For example, change TCPIP.H0004.HOSTS.SITEINFO to TCPIP.MVSA.HOSTS.SITEINFO and TCPIP.H0004.HOSTS.ADDRINFO to TCPIP.MVSA.HOSTS.ADDRINFO.

Testing the Site Table: After running MAKESITE, you can test the correctness of the *hlq*.HOSTS.ADDRINFO and *hlq*.HOSTS.SITEINFO data sets with TESTSITE.

TESTSITE Command

Use TESTSITE to verify that the *hlq*.HOSTS.ADDRINFO and *hlq*.HOSTS.SITEINFO data sets can correctly resolve the name of a host, gateway, or net.

Note: The TSO PING command uses the Pascal socket API, so VMCF must be started for the command to be successful. If VMCF is not started, an ABEND0D6 can occur.

Syntax

Parameters

None.

Examples

To test your HOSTS data sets, enter:

```
TESTSITE
```

When prompted for a name, enter the host, gateway or net name you want to verify.

When you have checked all the names in question, enter `QUIT` and press `ENTER`.

Usage Notes

TESTSITE gets its input from the *hlq*.HOSTS.ADDRINFO and *hlq*.HOSTS.SITEINFO data sets, where the HLQ is derived in this order:

- TSO userid or the TSO PROFILE PREFIX, if it is different from the userid
- The value specified with the DATASETPREFIX statement in PROFILE.TCPIP and TCPIP.DATA
- System default

Part 2. Configuring the Servers

Chapter 8. Configuring the SNALINK Environment 333
Understanding the SNALINK Environment 333
Configuration Process 334
Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP 334
Defining SNA DLC Links 334
Defining NCPROUTE and 3745 LAN Attachments 336
Step 2: Update the SNALINK Cataloged Procedure 337
SNALINK Cataloged Procedure (SNALPROC) 337
SNALINK Parameters 337
Step 3: Define the SNALINK Application to VTAM. 338
VTAM Considerations 339
Operating SNALINK. 339
Stopping and Starting SNALINK 339
Sample Console 340
Determining the DLC Connection Status Using NETSTAT DEVLINKS . . . 341
Controlling the SNALINK LU0 Interface with the MODIFY Command. 342
MODIFY Command—SNALINK LU0 343

Chapter 9. Configuring the SNALINK LU6.2 Interface 347
Configuration Process 347
Step 1: Specify DEVICE and LINK Statements in hlq.PROFILE.TCPIP . . . 347
Step 2: Update the SNALINK LU6.2 Cataloged Procedure 347
SNALINK LU6.2 Cataloged Procedure (LU62PROC). 348
Step 3: Define the SNALINK LU6.2 Application to VTAM 348
Step 4: Update the SNALINK LU6.2 Configuration Data Set 349
Summary of SNALINK LU6.2 Configuration Statements 349
Sample SNALINK LU6.2 Configuration Data Set (LU62CFG) 349
SNALINK LU6.2 Configuration Statements 350
Statement Syntax 350
Statement Ordering 350
BUFFERS Statement 351
DEST Statement 352
LINK Statement 353
TRACE Statement 354
VTAM Statement 355
MODIFY Command—SNALINK LU 6.2 356

Chapter 10. Configuring the X.25 NCP Packet Switching Interface (NPSI) Server 361
Configuration Process 361
Step 1: Specify X.25 Configuration Statements in hlq.PROFILE.TCPIP . . . 362
Step 2: Update the X.25 NPSI Cataloged Procedure. 362
X.25 NPSI Cataloged Procedure (X25PROC): 362
Step 3: Modify the X.25 NPSI Server Configuration Data Set 363
Summary of X.25 NPSI Server Configuration Statements 363
Sample X.25 NPSI Server Configuration Data Set (X25CONF) 363
Step 4: Define the X.25 NPSI Configuration 365
Step 5: Define the X.25 NPSI Application to VTAM 368
Step 6: Define VTAM Switched Circuits 368
Configuring Cross-Domain Resources 370
X.25 NPSI Server Configuration Statements. 370
Statement Syntax 370
ALTLINK Statement. 371

BUFFERS Statement 373
DEST Statement 374
FAST Statement 375
LINK Statement 376
OPTIONS Statement 377
TIMERS Statement 379
TRACE Statement 380
VTAM Statement 382
MODIFY Command—X.25 NPSI Server 383

Chapter 11. Configuring the OS/390 UNIX Telnet Server 387
Installation Information 387
Starting, Stopping, and Administration of OS/390 UNIX Telnet 388
otelnetd 392
SMF Record Handling 394
BPX.DAEMON Considerations 394

Chapter 12. Configuring the Telnet Server 395
Configuration Options 395
Customizing the TCPIP Cataloged Procedure 396
Customizing Translate Tables for 3270 DBCS Transform Mode 397
Customizing the VTAM Configuration Data Set 397
Customizing Statements in the PROFILE Data Set 398
Profile Modifications. 398
General Rules for PROFILE Statements 400
Specifying the Stand-Alone PORT Statement 402
Stand-Alone PORT and PORTRANGE Statements 403
Specifying the TELNETPARMS Statements in the PROFILE Data set . . . 404
BINARYLINEMODE Statement. 405
CLIENTAUTH 406
CODEPAGE Statement 408
DBCSTRACE Statement 409
DBCSTRANSFORM and TRANSFORM Statements 410
DISABLESGA Statement 411
ENCRYPTION and ENDENCRYPTION Statements 412
FULLDATATRACE Statement 414
INACTIVE Statement 415
NOTN3270E Statement 416
OLDSOLICITOR Statement 417
Port Designation Statements 418
PRTINACTIVE Statement 419
SCANINTERVAL and TIMEMARK Statements 420
SINGLEATTN Statement 421
SMFINIT and SMFTERM Statements 422
SSLTIMEOUT Statement 423
TESTMODE Statement 424
TKOSPECLU Statement 425
TIMEMARK Statement. 426
TRANSFORM Statement 427
WLMCLUSTERNAME Statement 428
Specifying BEGINVTAM Statements in the PROFILE Data Set 429
Specifying LOGMODE Using BEGINVTAM 430
VTAM Statements 431
ALLOWAPPL Statement 432
DEFAULTAPPL Statement 434
DEFAULTLUS and ENDDEFAULTLUS Statements 436

HNGROUP and ENDHNGROUP Statements . . . 437
INTERPTCP Statement . . . 438
IPGROUP and ENDIPGROUP Statements . . . 440
LINEMODEAPPL Statement . . . 441
LUGROUP and ENDLUGROUP Statements . . . 442
LUMAP Statement . . . 443
LUSESSIONPEND Statement . . . 445
MSG07 Statement . . . 446
PORT Statement . . . 447
PRTGROUP Statement . . . 448
PRTMAP Statement. . . . 449
QUEUESESSION Statement . . . 450
RESTRICTAPPL Statement . . . 451
TELNETDEVICE Statement . . . 453
USSTCP Statement . . . 455
Telnet PROFILE Example . . . 457
Explanation . . . 458
Configuration Considerations . . . 460
Choosing Telnet Solicitor or USSMSG Logon Panel . . . 460
Using the Default Telnet Solicitor Logon Panel . . . 461
Creating a Custom USSMSG Screen for Telnet . . . 461
Considerations for Using TN3270 Enhanced Support (TN3270E) . . . 480
TN3270E Device Pool Example . . . 482
Potential TN3270E Problem . . . 485
Logmode Considerations . . . 485
Telnet Device Name Parameters . . . 486
Selection Sequence Considerations for VTAM Applications and LUs . . . 486

Chapter 13. Managing the Telnet Server . . . 489
Operating the Telnet Server Using the VARY TCPIP Command Set . . . 489
VARY ACT Command . . . 490
VARY INACT Command . . . 491
VARY OBEYFILE Command . . . 492
VARY QUIESCE Command . . . 493
VARY RESUME Command . . . 494
VARY STOP Command . . . 495
Operating the Telnet Server Using the DISPLAY Command Set. . . . 496
The Profile Group Displays . . . 498
APPL Display Command . . . 499
DEFAULTS Display Command . . . 501
DEVICETYPE Display Command . . . 503
HNGROUP Display Command . . . 505
IPGROUP Display Command . . . 507
LINKNAME Display Command . . . 509
LUGROUP Display Command . . . 510
LUMAP Display Command . . . 512
PROFILE Display Command . . . 513
WHEREUSED Display Command . . . 515
The Connection Group Displays – CONNECTION Display Command . . . 517
The Port Group Displays – WLM Display Command . . . 520
The Server Group Displays - INACTLUS Display Command . . . 521

Chapter 14. Configuring the File Transfer Protocol (FTP) Server . . . 523
Configuration Process . . . 523
Step 1: Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS Information . 523
Step 2: Update ETC.SERVICES . . . 524

Step 3: Update the FTPD Cataloged Procedure 524
FTP Server Cataloged Procedure (FTPD). 524
Specifying the FTPD Parameters 525
Defining Optional Environment Variables for the FTP Server 527
Step 4: Specify FTP Configuration Statements in FTP.DATA 527
Summary of FTP Server Configuration Statements 528
Sample FTP Server Configuration Data Set (FTPSDATA) 530
Specifying Attributes for New MVS Data Sets 532
Step 5: Configure the FTP Server for SMF (Optional) 533
Summary of FTP Server SMF Statements 534
FTP Server SMF User Exit 535
Example FTPSMFEX User Exit 535
Step 6: Configure the User-Written Exits (Optional) 536
The FTCHKIP User Exit 537
The FTCHKPWD User Exit 537
The FTCHKCMD User Exit 538
The FTCHKJES User Exit 538
Step 7: Specify Configuration Statements in TCPIP.DATA 539
Summary of FTP Server TCPIP.DATA Statements. 539
Step 8: Install the SQL Query Function (Optional) and Access the DB2 Modules 539
Accessing DB2 Modules 541
Step 9: Update /etc/syslog.conf for the FTP Server 542
Implementing an Anonymous User 542
Identified User. 542
Anonymous User. 544
Security Considerations for the FTP Server 545
FTP.DATA Data Set Statements 546
ANONYMOUS Statement 547
ASATRANS Statement. 549
AUTOMOUNT Statement. 550
AUTORECALL Statement 551
AUTOTAPEMOUNT Statement 552
BLKSIZE Statement. 553
BUFNO Statement 554
CCXLATE Statement 555
CHKPTINT Statement 556
CONDDISP Statement. 557
CTRLCONN Statement 558
DATACLASS Statement 559
DB2 Statement 561
DB2PLAN Statement 562
DCBDSN Statement 563
DEST Statement 565
DIRECTORY Statement 566
DIRECTORYMODE Statement. 567
FILETYPE Statement 568
INACTIVE Statement 569
JESLRECL Statement 570
JESOUTGETTO Statement 571
JESRECFM Statement 572
LRECL Statement 573
MGMTCLASS Statement 574
MIGRATEVOL Statement. 575
PRIMARY Statement 576
QUOTESOVERRIDE Statement 577

RDW Statement . . . 578
RECFM Statement . . . 579
RETPD Statement . . . 580
SBDATACONN Statement . . . 581
SECONDARY Statement . . . 582
SMF Statement . . . 583
SMFAPPE Statement . . . 585
SMFDEL Statement . . . 586
SMFEXIT Statement . . . 587
SMFJES Statement . . . 588
SMFLOGN Statement . . . 589
SMFREN Statement . . . 590
SMFRETR Statement . . . 591
SMFSQL Statement . . . 592
SMFSTOR Statement . . . 593
SPACETYPE Statement . . . 594
SPREAD Statement . . . 595
SQLCOL Statement . . . 596
STARTDIRECTORY Statement . . . 597
STORCLASS Statement . . . 598
TRACE Statement . . . 599
TRAILINGBLANKS Statement . . . 600
UCSHOSTCS Statement . . . 601
UCSSUB Statement . . . 602
UCSTRUNC Statement . . . 603
UMASK Statement . . . 604
UNITNAME Statement . . . 605
VOLUME Statement . . . 606
WLMCLUSTERNAME Statement . . . 607
WRAPRECORD Statement . . . 608
XLATE Statement . . . 609
Starting, Stopping, and Tracing the FTP Server . . . 610
Starting the FTP Server . . . 610
Starting FTP as a Started Task . . . 610
Starting FTP from the OS/390 Shell . . . 610
Starting FTP Automatically . . . 610
FTP Server Exit Codes . . . 611
Stopping the FTP Server . . . 611
Tracing the FTP Server . . . 611
Syntax . . . 612
Parameters . . . 612
Examples . . . 612
Related Topics . . . 613
FTP Code Page Conversion . . . 613
Code Page Conversions for the Control Connection . . . 613
Priority . . . 613
Code Page Conversions for the Data Connection . . . 613
Priority . . . 614

Chapter 15. Configuring the Trivial File Transfer Protocol Server . . . 615
Considerations for OS/390 . . . 615

Chapter 16. Configuring the TIMED Daemon . . . 619
Starting TIMED from OS/390 . . . 619
Starting TIMED as a Procedure . . . 619

Chapter 17. Configuring the OS/390 Unix Remote Execution Server . . . 621
Installation Information . . . 621
HFS Files for OS/390 UNIX REXECD . . . 621
HFS Files for OS/390 UNIX RSHD . . . 622
Setting up the OS/390 UNIX RSHD Installation Exit . . . 622
OS/390 UNIX REXECD Command (orexecd) . . . 623
OS/390 UNIX RSHD Command (orshd) . . . 623

Chapter 18. Configuring the Remote Execution Server . . . 625
Configuration Process . . . 625
Step 1: Specify the AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP . . . 625
Step 2: Determine Whether Remote Execution Client Will Send REXEC or RSH Commands . . . 626
Step 3: Permit Remote Users to Access MVS Resources . . . 626
Step 4: Update the Remote Execution Cataloged Procedure . . . 627
Remote Execution Cataloged Procedure (RXPROC) . . . 627
Specify the Remote Execution Server Parameters . . . 628
Step 5: Create a User Exit Routine (Optional) . . . 629
Modifying the Remote Execution Server Operating Parameters . . . 632
MODIFY Command—Remote Execution Server . . . 633

Chapter 19. Configuring the SMTP Server . . . 635
Configuration Process . . . 635
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP . . . 635
Step 2: Update the SMTP Cataloged Procedure . . . 636
SMTP Cataloged Procedure (SMTPPROC) . . . 636
Step 3: Customize the System CLIST and Modify PARMLIB Data Sets . . . 637
Step 4: Customize the SMTP Mail Headers (Optional) . . . 638
The SMTP Rules Data Set . . . 639
Predefined Keywords within the SMTP Rules . . . 644
Default SMTP Rules . . . 645
Examples of Header Rewrite Rules . . . 646
Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional) . . . 647
Step 6: Specify Configuration Statements in SMTP Configuration Data Set . . . 648
Summary of SMTP Configuration Statements . . . 648
Sample SMTP Configuration Data Set (SMTPCONF) . . . 650
Step 7: Create an SMTP Security Table (Optional) . . . 653
SMTP Security Data Set Examples . . . 653
Rejected Mail Examples . . . 654
Step 8: Enable SMTP Domain Name Resolution . . . 655
SMTP Configuration Data Set Statements . . . 656
ALTNJEDOMAIN Statement . . . 657
ALTTCPHOSTNAME Statement . . . 658
BADSPOOLFILEID Statement . . . 659
DBCS Statement . . . 660
DEBUG Statement . . . 663
FINISHOPEN Statement . . . 664
GATEWAY Statement . . . 665
INACTIVE Statement . . . 667
IPMAILERADDRESS Statement . . . 668
LISTENONADDRESS Statement . . . 669
LOCALCLASS Statement . . . 670
LOCALFORMAT Statement . . . 671
LOG Statement . . . 672
MAILER Statement . . . 673

MAILFILEDSPREFIX Statement 675
MAILFILEUNIT Statement 676
MAILFILEVOLUME Statement 677
MAXMAILBYTES Statement 678
NJECLASS Statement 679
NJEDOMAIN Statement 680
NJEFORMAT Statement 681
NJENODENAME Statement 682
NOLOG Statement 683
OUTBOUNDOPENLIMIT Statement 684
PORT Statement 685
POSTMASTER Statement 686
RCPTRESPONSEDELAY Statement 687
RESOLVERRETRYINT Statement 688
RESTRICT Statement 689
RETRYAGE Statement 691
RETRYINT Statement 692
REWRITE822HEADER Statement 693
SECURE Statement 694
SMSGAUTHLIST Statement 695
SPOOLPOLLINTERVAL Statement 696
TEMPERRORRETRIES Statement 697
TIMEZONE Statement 698
WARNINGAGE Statement 699
Operating the SMTP Server 700
Using SMSG with SMTP 700
Forcing Re-resolution of Queued Mail 700

Chapter 20. Configuring OS/390 UNIX sendmail and Popper 701
Overview 701
Configuring OS/390 UNIX sendmail 703
The sendmail Samples Directory 703
Creating the Configuration File 704
Retrieving the m4 Preprocessor 705
Creating the .mc File 705
Building the Configuration File 706
Creating the Aliases File 706
Configuration Hints and Tips 707
Configuring Popper 708

Chapter 21. Configuring the Bind-Based Domain Name System (DNS) . . 709
DNS and BIND Overview 709
Domain Name Servers 710
Master Servers 711
Caching-Only Servers 711
Forwarders 711
Slaves 712
Recommended Reading 712
Setting Up and Running the Name Server 713
Migrating to the Name Server 713
Configuring the Name Server 714
Step 1. Specify Stack Affinity (Multiple Stack Environment) 714
Step 2. Specify Port Ownership 714
Step 3. Update the Name Server Start Procedure 714
Step 4. Create the Domain Data Files (Primary Name Server Only) . . . 715
Step 5. Create the Hints (Root Server) File 716

Step 6. Create the Loopback File . . . 716
Step 7. Create the Boot File. . . . 716
Step 8. Configure the Start Process . . . 717
Configuring Host Resolvers: Name Server Considerations . . . 718
Creating the Syslog File . . . 718
Querying Name Servers . . . 718
onslookup/nslookup Command. . . . 719
Entering the Interactive Mode . . . 719
Entering the Command Line Mode . . . 719
onslookup/nslookup Configuration . . . 719
Configuring Host Resolvers: onslookup/nslookup Considerations . . . 721
Diagnosing Problems . . . 722
Checking Messages Sent to the Operators Console . . . 722
Checking the Syslog Messages . . . 723
Using Name Server Signals to Diagnose Problems . . . 723
Using onslookup/nslookup to Diagnose Problems . . . 723
Connection Optimization in a Sysplex Domain . . . 723
Overview. . . . 724
Registration. . . . 724
Name Resolution. . . . 725
Generated Names vs. Statically Defined Names . . . 726
Usage Considerations in a Connection Optimized Sysplex . . . 729
Caching Issues . . . 731
Configuring a Sysplex Domain for Connection Optimization . . . 731
Step 1: Identify Server Applications . . . 731
Step 2: Configure Server Applications for WLM Registration . . . 732
Step 3: Choose Sysplex Name and Identify Name Servers . . . 733
Step 4: Update Parent Domain Name Server . . . 733
Step 5: Configure the Sysplex Name Servers . . . 734
Step 6: Configure Client Applications . . . 736
Step 7: Configure WLM in Goal Mode . . . 736
Registering Your Own Applications . . . 736
Dynamic IP . . . 737
Overview. . . . 737
Objectives and Customer Benefits of Dynamic IP . . . 738
Dynamic IP Components . . . 738
Administering Dynamic Domains . . . 739
Migrating an Existing DNS Configuration to Dynamic IP . . . 740
RSA Encryption . . . 740
Secret Key Cryptography. . . . 740
Public Key Cryptography . . . 740
Encryption and Authentication . . . 740
Hash Functions . . . 741
The RSA Encryption Standard . . . 741
Configuring the DHCP Server for OS/390. . . . 742
How Does DHCP Work? . . . 743
Setting Up a DHCP Network . . . 745
Changing the DHCP Configuration File. . . . 749
Editing Tips . . . 750
DHCP Server Statements . . . 750
A Sample DHCP Server Configuration Files . . . 759
Specifying DHCP Options . . . 760
Hardware Types . . . 771
Configuring DHCP Server as DDNS Client Proxy . . . 772
Enabling Dynamic A Record Updates . . . 772
Specifying the Keyword for A Record Updates . . . 773

Releasing a Client A Record 774
Specifying the Keyword for PTR Record Updates 775
Releasing a Client PTR Record 776
Defining DHCP Proxy Authority 776
Defining DDNS Key Files for the PTR Record 776
Defining DDNS Key Files for the A Record 777
Configuring the BINL Server 778
Configuring a DDNS Server 783
Creating a New DDNS Server Configuration 783
Configuring for Presecured Mode Operation 786
Control Entries, Resource Records, and Special Characters 787
Control Entries 787
Resource Records 787
Special Characters 791
Boot File Directives 792
Sample Files 794
Sample Forward Domain Data File 794
Sample Reverse Domain Data File 795
Sample Hints (Root Server) File 795
Sample Loopback File 797
Sample Boot File 797
The named Daemon 799
onslookup/nslookup—Querying A Name Server in Command Mode 803
onslookup/nslookup—Issuing Multiple Queries in Interactive Mode 805
nsupdate Command 818
nsupdate Subcommands 820
nsupdate Examples 820
How an Administrator Removes and Locks Out a Host Name 820
How an Administrator Creates an Alias for the Dynamic Zone 821
Return Codes 821

Chapter 22. Configuring Simple Network Management Protocol (SNMP) . . 823
Processing SNMP Request 823
Deciding on SNMP Security Needs 824
Step 1: Configure the SNMP Agent (OSNMPD) 825
Provide TCP/IP Profile Statements 826
Provide Community-Based Security and Trap Destination Information . . . 827
Provide Community Name Information 827
Provide Trap Destination Information 828
Provide Community-Based and User-Based Security and Trap Destination Information 830
SNMPD.CONF File 830
SNMPD.BOOTS 845
SNMPD.BOOTS Search Order 845
SNMPD.BOOTS Statement Syntax 846
Creating User Keys 847
Provide MIB Object Configuration Information 850
OSNMPD.DATA Search Order 850
OSNMPD.DATA Statement Syntax 851
OSNMPD.DATA Example 851
Start the SNMP Agent (OSNMPD) 852
OSNMPD Parameters 852
Sample JCL Procedure for Starting OSNMPD from MVS 853
Command for Starting OSNMPD from OS/390 UNIX 854
Step 2: Configure the SNMP Commands 855
Configure the NetView SNMP (SNMP) Command 855

Configure the SNMP Query Engine 855
Configure NetView as an SNMP Monitor 858
Configure the OS/390 UNIX SNMP (osnmp) Command 861
Provide osnmp Configuration Information 861
Provide User MIB Object Information 865
Step 3: Configure the SNMP Subagents 866
Step 4: Configure the ATM Open Systems Adapter 2 (ATM OSA-2) Support . . 867
OSA/SF Prerequisites 868
Required TCP/IP Profile Statements 869
Multiple TCP/IP Instances Considerations 869
Subagent Connection to OSA/SF 869

Chapter 23. Configuring the Kerberos Server 871
Configuration Process 871
Step 1: Add PORT Statements to the hlq.PROFILE.TCPIP Data Set 871
Step 2: Update the Authentication Server Cataloged Procedure. 872
Kerberos Cataloged Procedure (MVSKERB). 872
Step 3: Update the Remote DBA Server Cataloged Procedure 872
ADM@SERV Cataloged Procedure (ADM@SERV) 872
Step 4: Define the Kerberos Services in hlq.ETC.SERVICES 873
Step 5: Create and Update the Kerberos System Data Sets 873
Kerberos Configuration Data Set (KRBCONF) 873
Kerberos Remote DBA Authorization Data Sets 874
Step 6: Authorize Kerberos Servers 875
Step 7: Build the Kerberos Database 875
Step 8: Store the Master Key 875
Step 9: Start the Kerberos Servers 875
Start the Kerberos Authentication Server 876
Start the Kerberos Remote DBA Server 876
Verifying the Kerberos Configuration. 877
Step 1: Set Up the Environment 878
Step 2: Register the Sample Service and the User 878
Step 3: Generate the Key Data Set for the Sample Service 879
Step 4: Transfer the Service Key Data Set to the Server 879
Step 5: Start the Sample Server 879
Step 6: Get the Initial Ticket 880
Step 7: Run the Sample Client Program 880
Setting Up a Service or Client Application. 880
Setting Up a Service Application 880
Setting Up a Client Application 881
Administrative Commands for the Kerberos Database 881
EXT@SRVT Command 883
KADMIN Command 884
KDB@DEST Command 886
KDB@EDIT Command 887
KDB@INIT Command 889
KDB@UTIL Command. 890
KSTASH Command 891

Chapter 24. Configuring the Remote Print Server (LPD) 893
Configuration Process 893
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP. . 893
Step 2: Update the LPD Server Cataloged Procedure 894
LPD Server Cataloged Procedure (LPSPROC) 894
Specifying LPD Server Parameters 895
Alternative Method for Specifying the LPD Configuration File 895

Step 3: Update the LPD Server Configuration Data Set. 896
Summary of LPD Server Configuration Statements 896
Sample LPD Server Configuration Data Set (LPDDATA) 897
Step 4: Create a Banner Page (Optional) 898
Statements for the LPD Server Configuration Data Set 899
Syntax Rules 899
DEBUG Statement 900
JOBPACING Statement 901
OBEY Statement 902
SERVICE Statement 903
STEPLIMIT Statement 913
UNIT Statement 914
VOLUME Statement 915
Operating the LPD Server Using the SMSG Interface 916

Chapter 25. Configuring the OROUTED Server 917
Understanding OROUTED 917
Routing Information Protocol (RIP) 918
RIP Version 2 919
OROUTED Miscellaneous Features 920
RIP Input/Output Filters 920
RIP Routes 921
OROUTED Gateways 921
Passive RIP Routes. 921
External RIP Routes 922
Active Gateways 922
OROUTED Gateways Summary 922
Configuration Process 923
Step 1: Configure the OROUTED Profile 923
Step 2: Update Configuration Statements in PROFILE.TCPIP 925
Step 3: Update the Resolver Configuration File. 926
Step 4: Update the OROUTED Cataloged Procedure (Optional) 926
OROUTED Cataloged Procedure 927
Step 5: Specify the OROUTED Port Number in the SERVICES File 928
Step 6: Configure the Gateways File or Data Set (Optional) 928
Syntax Rules 929
Step 7: Configure and Start syslogd 934
Step 8: RACF-Authorize User IDs 934
OROUTED Parameters 934
Specifying Parameters 936
Starting OROUTED 937
Configuration Examples 937
Configuring a Passive Route 938
Configuring an External Route 939
Configuring an Active Gateway 939
Configuring a Point-to-Point Link 940
Configuring a Default Route 940
Configuring ORouteD with Enterprise Extender 940
Configuring OROUTED with VIPA 941
Configuring OROUTED to Split Traffic with VIPA 941
Configuring a Backup TCP/IP Stack with VIPA 943
Restoring a Primary OS/390 TCP/IP Stack with VIPA 944
Controlling OROUTED with the MODIFY Command 945

Chapter 26. Configuring OMPROUTE 949
Understanding OMPROUTE. 949

Open Shortest Path First (OSPF) 951
Routing Information Protocol (RIP) 952
The Configuration Process 953
Step 1: Create the OMPROUTE Configuration File 953
Step 2: Reserve the RIP UDP Port (If Using the RIP Protocol) 954
Step 3: Update the Resolver Configuration File. 954
Step 4: Update the OMPROUTE Cataloged Procedure (Optional) 955
Step 5: Specify the RIP UDP Port Number in the SERVICES File or Data Set (If Using the RIP Protocol) 955
Step 6: RACF-Authorize User IDs for Starting OMPROUTE 956
Step 7: Start syslogd 956
Step 8: Update the OMPROUTE Environment Variables (Optional) 956
Step 9: Create Static Routes (Optional) 956
Starting OMPROUTE 957
OMPROUTE Parameters 958
Controlling OMPROUTE with the MODIFY Command 958
Stopping OMPROUTE 959
Rereading the Configuration File 959
Enabling or Disabling the OMPROUTE Subagent 959
Changing the Cost of OSPF Links 960
Controlling OMPROUTE Tracing 960
OMPROUTE Configuration File 960
OSPF Configuration Statements 960
AREA 962
COMPARISON 963
OSPF_INTERFACE 964
VIRTUAL_LINK 969
ROUTERID 971
AS_BOUNDARY_ROUTING 972
RANGE 974
DEMAND_CIRCUIT 975
RIP Configuration Statements 976
ORIGINATE_RIP_DEFAULT 977
RIP_INTERFACE 978
ACCEPT_RIP_ROUTE 984
FILTER 985
Send_Only 986
Common Configuration Statements 987
DEFAULT_ROUTE 988
INTERFACE 989
ROUTESA_CONFIG Statement 991
Using OMPROUTE DISPLAY Commands. 993
All OSPF Configuration Information 993
Configured OSPF Areas and Ranges 995
Configured OSPF Interfaces 995
Configured OSPF Non-Broadcast, Multi-Access Networks 996
Configured OSPF Virtual Links. 996
Configured OSPF Neighbors 997
OSPF Link State Advertisement 997
OSPF Area Statistics and Parameters 999
OSPF External Advertisements 1000
OSPF Area Link State Database 1000
OSPF Interface Statistics and Parameters 1001
OSPF Neighbor Statistics and Parameters 1003
OSPF Router Routes 1005
OSPF Link State Database Statistics 1006

OSPF Routing Protocol Statistics 1006
OMPROUTE Routing Table 1008
Route Expansion Information 1009
RIP Configuration Information 1010
Configured RIP Interfaces 1011
RIP Routes to Be Accepted 1011
RIP Interface Statistics and Parameters 1012
Global RIP Filters 1013
Configuring Interfaces to OMPROUTE 1014
Configuring Point-to-Point Interfaces. 1015
Configuring MPC, XCF and IUT SameHost Interfaces 1015
Configuring Non-Broadcast Network Interfaces 1016
Configuring Broadcast Network Interfaces 1017
Configuring VIPA Interfaces 1017
Special Considerations for Dynamic VIPA 1017

Chapter 27. Using OSPF 1021
OSPF Routing Summary 1021
Designated Router 1022
Configuring OSPF 1022
Setting OSPF Router IDs 1023
Defining Backbone and Attached OSPF Areas 1023
Setting OSPF Interfaces 1026
OSPF Domain Topology 1026
Costs for OSPF Links 1027
Changing the Cost of OSPF Links 1027
Interactions between Neighbor Routers 1027
Setting Non-Broadcast Network Interface Parameters 1028
Configuring Wide Area Subnetworks. 1028
Configuring Point-to-Multipoint Subnetworks 1029
Configuring NBMA Subnetworks 1029
Enabling AS Boundary Routing 1029
Configuring OSPF over ATM 1030
Configuring OSPF over ATM Native 1030
Other Configuration Tasks 1030
Setting Virtual Links 1030
Configuring for Routing Protocol Comparisons 1031
Demand Circuit 1032
Request Hello Suppression 1033
PP_Poll_Interval 1033
Converting from RIP to OSPF 1033

Chapter 28. Configuring the NCPROUTE Server 1035
Understanding NCPROUTE 1035
Environment 1036
Server Requirements 1037
NCPROUTE Operation 1037
NCPROUTE Gateways 1037
NCPROUTE Active Gateways 1038
NCPROUTE Gateways Summary. 1039
RIP Input/Output Filters 1039
Configuration Process 1039
Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP 1040
Step 2: Configure VTAM and SNALINK Applications 1041
Step 3: Configure the IP over CDLC DEVICE and LINK Statements . . . 1042
Step 4: Update the NCPROUTE Cataloged Procedure 1042

NCPROUTE Cataloged Procedure (NCPROUT) 1042
Specifying the NCPROUTE Parameters 1044
Step 5: Update hlq.ETC.SERVICES 1045
Step 6: Configure the Host-Dependent NCP Clients 1045
Generating the Routing Information Tables 1045
Determining the Gateway Route Table Name 1046
NCST Session Interface Definition 1046
Channel PU Interface Definition 1047
NCP Host Interface Definition 1047
Step 7: Configure the NCPROUTE Profile Data Set 1049
Step 8: Configure the NCPROUTE Gateways Data Set (Optional) 1051
Syntax Rules 1052
Configuring a Passive Route 1058
Configuring an External Route 1059
Configuring an Active Gateway 1059
Configuring a Default Route 1060
Configuration Examples 1060
Step 9: Define a Directly Connected Host Route for the NCST Session . . 1061
Controlling the NCPROUTE Address Space with the MODIFY Command . 1061
MODIFY Command—NCPROUTE Address Space 1062

Chapter 29. Configuring the OS/390 Unix PORTMAP Address Space . . 1065
Configuration Process 1065
Step 1: Specify the PORT Statements in PROFILE.TCPIP 1065
Step 2: Update the PORTMAP Cataloged Procedure 1065
PORTMAP Cataloged Procedure (OPORTPRC) 1065
Starting the PORTMAP Address Space 1066

Chapter 30. Configuring the PORTMAP Address Space 1067
Configuration Process 1067
Step 1: Specify the AUTOLOG and PORT Statements in PROFILE.TCPIP 1067
Step 2: Update the PORTMAP Cataloged Procedure 1068
PORTMAP Cataloged Procedure (PORTPROC) 1068
Step 3: Define the Data Set for Well-Known Procedure Names 1068
Starting the PORTMAP Address Space 1071

Chapter 31. Configuring the NCS Interface 1073
Understanding the NCS Interface 1073
Understanding the LLBD Server 1073
Understanding the NRGLBD Server 1073
Configuration Process 1074
Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP. 1074
Step 2: Update the NRGLBD Cataloged Procedure 1074
NRGLBD Cataloged Procedure (NRGLBD) 1075
Step 3: Update the LLBD Cataloged Procedure 1075
LLBD Cataloged Procedure (LLBD) 1075

Chapter 32. Configuring the Network Database (NDB) System 1077
Configuration Process 1077
Step 1: Update the NDB Setup Sample Job 1077
NDB Set Up Cataloged Procedure (NDBSETUP) 1078
Step 2: Run the NDB Setup Job 1079
Step 3: Update and Install the DB2 Sample Connection Exit Routine. . . 1080
DSN3SATH ASSEMBLE Modifications for NDB 1080
Step 4: Update the PORTS Cataloged Procedure 1081
PORTS Cataloged Procedure (PORTSPRC). 1081

Step 5: Update the PORTC Cataloged Procedure 1082
Specifying PORTC Parameters 1082
PORTC Cataloged Procedure (PORTCPRC) 1082
Running Multiple PORTC Procedures 1083
Step 6: Create the NDB Clients 1084
Creating an NDB Client in the AIX Environment 1084
Creating an NDB Client in the SUN UNIX Environment 1085
Creating an NDB Client in the OS/2 Environment 1085
Creating an NDB Client in the DOS Environment 1086
Creating an NDB Client in the VM Environment 1087
Creating an NDB Client in the MVS Environment 1088
Starting NDB 1089

Chapter 33. Configuring the Miscellaneous (MISC) Server 1091
Understanding the MISC Server 1091
Discard Protocol 1091
Echo Protocol 1091
Character Generator Protocol 1091
Configuration Process 1092
Step 1: Specify AUTOLOG and PORT Statements in PROFILE.TCPIP . . 1092
Step 2: Update the MISC Server Cataloged Procedure (MISCSERV) . . 1093
MISC Server Cataloged Procedure (MISCSERV) 1093
Specifying the MISC Server Parameters 1094

Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent 1095
Configuring the Service Policy Agent 1095
Overview 1095
The Policy Profile Control Block (ProfileData) 1096
The Policy Configuration File 1096
TcpImage Statement 1097
ReadFromDirectory Statement 1098
LogLevel Statement 1100
SetSubnetPrioTosMask Statement 1101
ServiceCategories Statement 1103
ServicePolicyRules Statement 1106
Starting the Service Policy Agent 1109
Starting Pagent from the OS/390 Shell 1109
Starting Pagent as a Started Task 1110
Stopping the Service Policy Agent 1111
Configuring the SLA Subagent 1111
SLA Subagent Performance Monitoring 1111
Creating Monitor Table Entries and Enabling SNMP Traps 1112
Creating the Monitor Table Index 1113
Starting the SLA Subagent 1117
Starting the SLA Subagent from the OS/390 Shell 1117
Starting the SLA Subagent as a Started Task 1118
Stopping the SLA Subagent 1119
Controlling the SLA Subagent with the Modify Command 1119
Security Information 1120

Chapter 35. Configuring the RSVP Agent 1121
Description of RSVP 1121
Configuring the RSVP Agent 1122
The RSVP Configuration File 1122
LogLevel Statement 1123

TcpImage Statement 1124
Interface Statement 1125
RSVP Statement 1127
Starting the RSVP Agent 1129
Starting RSVP from the OS/390 Shell 1129
Starting RSVP as a Started Task 1129
Stopping RSVP 1130
Security Information 1130

Chapter 8. Configuring the SNALINK Environment

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

SNALINK allows TCP/IP to send and receive packets using SNA sessions instead of dedicating physical network hardware (such as a channel-to-channel adapter or channel connection to a 3745/46 Communication Controller).

SNALINK allows an installation to multiplex SNA and IP traffic over the same I/O subchannels, rather than requiring separate subchannels dedicated to VTAM and TCP/IP. While such multiplexing capability may be desirable at some installations, the native TCP/IP CTC and 3745/46 device drivers will likely outperform SNALINK connections. Interaction with the SNALINK address space is very CPU-intensive, and is not required with the native TCP/IP CTC and 3745/46 device drivers. (See “DEVICE and LINK Statement—CTC Devices” on page 155 and “DEVICE and LINK Statement—3745/46 Channel DLC Devices” on page 190 for configuration information.) It is therefore important to weigh the multiplexing capability that SNALINK provides against its performance cost, in determining whether to use SNALINK or the native TCP/IP CTC or 3745/46 device drivers.

Notes:

1. The IUCV keyword remains for migration purposes and is identical to SAMEHOST.
2. Prior to NCP V7R3, NCP did not support native IP transmission of the transport PDUs associated with RIP traffic, across the channel. NCP expects these PDUs to be carried in SNA Frames. SNALINK is therefore still required at installations where dynamic routing is performed with the NCP (via NCPROUTE). See “DEVICE and LINK Statement—3745/46 Channel DLC Devices” on page 190 for more information.

Understanding the SNALINK Environment

The SNALINK environment interfaces between the TCP/IP environment’s SNAIUCV driver and the customer’s SNA network. SNALINK communicates with one or more instances of SNALINK at remote nodes, using the SNA LU type 0 protocol. See Figure 13 on page 334 for a description of the SNALINK environment interfaces.

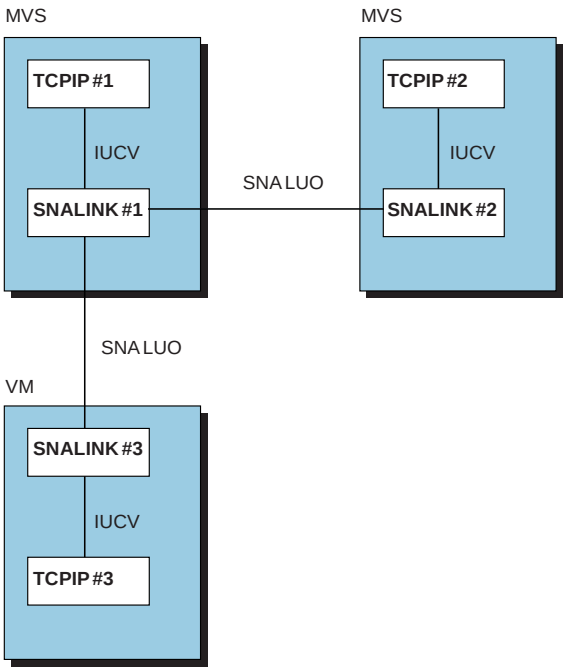


Figure 13. SNALINK Environment Interfaces

Each SNALINK environment can communicate with up to 9999 SNALINKs simultaneously. The number of connections is determined by the parameters you pass to the SNALINK cataloged procedure. The default is 6 sessions running in dual mode for a total of 3 SNALINKs.

- When operating in single mode, SNALINK opens one full duplex session.
- When operating in dual mode, SNALINK opens two System Network Architecture (SNA) sessions for each remote logical unit (LU) with which it communicates, one for sending and one for receiving.

Configuration Process

Steps to configure SNALINK:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP
2. Update the SNALINK cataloged procedure
3. Define the SNALINK application to VTAM

For information about starting and stopping SNALINK and controlling the SNALINK interface, see “Operating SNALINK” on page 339.

Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP

The following sections describe the changes you must make to your TCPIP address space configuration data set (*hlq*.PROFILE.TCPIP).

Defining SNA DLC Links

SNA DLC links are point-to-point and require DEVICE and LINK statements in the configuration data set. The DLC link constitutes a separate network, even though it includes only two hosts. To define a link, each host to which the DLC link is attached requires:

- A pair of SNA LU0 DEVICE and LINK statements
- A HOME statement
- A BSDROUTINGPARMS statement or a GATEWAY statement

SNA DLC links are defined in one of two ways:

- By unique network or subnetwork numbers, if the hosts to which they connect are not attached to other networks.
- By the IP address of the hosts to which they connect, if the hosts are attached to other networks.

You usually have to assign a unique network or subnetwork number to the SNALINK. If the link connects 2 hosts that also have other networks attached to them, the DLC link does not need its own subnetwork number. Figure 14 illustrates how to define an SNA DLC link if the 2 hosts are connected to other networks in the following way:

- Host A and Host B are connected by SNA DLC
- Host A is also connected to a token ring, 193.1.1
- Host B is also connected to a token ring, 193.1.2
- Host A's home address on its token ring is 193.1.1.1
- Host B's home address on its token ring is 193.1.2.1

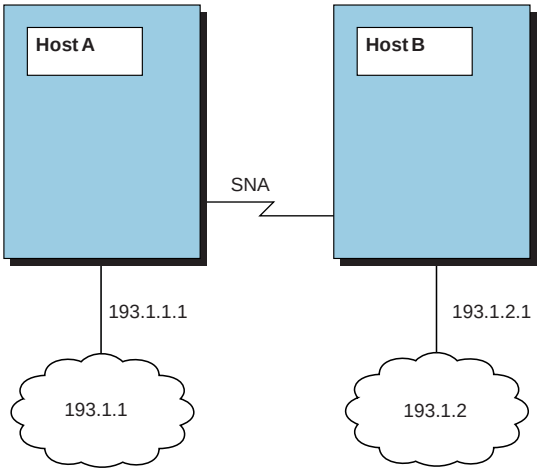


Figure 14. SNA DLC Link

Host A's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE  LCS1 LCS BA0
LINK  TR1  IBMTR 0 LCS1
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.1.1  TR1
    193.1.1.2  SNAIUCV1

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.1.0      =          TR1         2000        0
    193.1.2.0      =          SNAIUCV1    2000        0
```

Host B's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE LCS2 LCS BE0
LINK  TR1 IBMTR 0 LCS2
DEVICE SNALU0 SNAIUCV SNALINK LU000001 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.2.1  TR1
    193.1.2.2  SNAIUCV1

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.2.0      =          TR1         2000        0
    193.1.1.0      =          SNAIUCV1    2000        0
```

Notes:

1. The *lu_name* must be different on each host. In the example, the *lu_name* for Host A is lu000000. The *lu_name* for Host B is lu000001.
2. In the example, the *lu_name* for Host A is the remote or partner LU.

Hosts A and B are addressed by their token-ring home addresses, even if the packets reach them through the SNA DLC link.

If Host B had no other network attached to it, you would have to assign a separate subnetwork number to the SNA DLC link. Even in this case, Host A does not need a separate home address for its SNA link, because it can be addressed by its token-ring home address. Host B's only home address is the home address for the SNA link.

Note: If you plan to run a network-monitoring protocol that requires each subnet to have its own subnet number, you can assign a separate subnet network number to the DLC link.

For additional information on configuring these links, see "DEVICE and LINK Statement—SNA LU 6.2 Links" on page 182.

Defining NCPROUTE and 3745 LAN Attachments

If your TCP/IP configuration supports NCPROUTE or 3745 Communications Controller Ethernet or token-ring links, you need to match the *lu_name* on the DEVICE statement to the LU statement in NCST section of your NCP generation.

The following example shows the LU name A04TOLU1 defined in the *hlq*.PROFILE.TCPIP DEVICE statements and in the NCP generation.

```
DEVICE  SNA1LINK SNAIUCV SNALINK A04TOLU1 SNAL1STC
LINK  SNALINK SAMEHOST 1 SNA1LINK

HOME
    9.67.116.66  SNALINK

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    9.67.116.65   =          SNALINK     2000        HOST

START SNA1LINK
```

```
*************************************************************************
*               NCST IP INTERFACES**
*************************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),LIN*
              ECB=CXSXLNK
```

```
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000S
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSXL0000,CXSS0*
               000),LOCADDR=1
************************************************************************
```

In addition, the remote LU name SNALKLU1 in the NCP generation must match the APPLID in the SNALINK cataloged procedure parameters and in the VTAM APPL definition.

```
//SNALINK  PROC MODULE=SNALINK,TCPID='TCPV3',APPLID='SNALKLU1'
//SNALINK  EXEC PGM=&MODULE<REGION=$4096K,TIME=1440,
           PARM='&TCPID &APPLID C7 6 0003 SINGLE'
```

For additional information on configuring these links, see “DEVICE and LINK Statement—Virtual Devices” on page 188.

Step 2: Update the SNALINK Cataloged Procedure

Update the SNALINK cataloged procedure by copying the sample in *hlq*.SEZAINST(SNALPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify SNALINK parameters and change the DD statements, as required.

SNALINK Cataloged Procedure (SNALPROC)

```
//SNALINK  PROC MODULE=SNALINK,TCP='TCPIP',APPLID='APPLID'
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEB01U
//*
//*        5655-HAL (C) Copyright IBM Corp. 1989, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//*
//SNALINK  EXEC PGM=&MODULE,REGION=2048K,TIME=1440,PARM='&TCP &APPLID'
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task. The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

SNALINK Parameters

The system parameters required by SNALINK are passed by the PARM parameter on the EXEC statement of the SNALINK cataloged procedure. The parameters are order-dependent and appear in the following list:

DEBUG

Enables detailed tracing into an internal buffer. If specified, it must be the first parameter in the list.

TCP='tcpid'
Specifes the name of the TCPIP address space, in quotes.

APPLID='*applid***'**
Specifies the name of the VTAM application (LU name), in quotes, that SNALINK uses for this system.

max_ru_size
This parameter is optional, and is the maximum RU size in hexadecimal. Set *max_ru_size* to specify the maximum request or response unit (RU) size that SNALINK sends. The value is of the form *mn* , where *m* is between 8 and F, and *n* is between 0 and F. The corresponding maximum RU size is $m2^n$ (*m* multiplied by 2 to the power of *n*). Use the largest size that works on your SNA network, to provide the best performance and the least overhead. See *OS/390 eNetwork Communications Server: SNA Programming* for more information about this parameter as well as "VTAM Considerations" on page 339.

max_session
The maximum number of sessions; a decimal value from 1 to 9999. The default value is 6. To use different values for *max_session*, you also have to specify the *max_ru_size*.

Retry The delay time for VTAM to retry sense codes. It has the following format: *hhmm* Where:

hh Hours 0–24

mm Minutes 0–59

For example:

- 0005 is a 5-minute delay
- 0200 is a 2-hour delay
- 1030 is a 10-hour and 30-minute delay

The default delay is 15 minutes and the maximum retry is 24 hours. To use a different retry interval, you must specify both *max_ru_size* and *max_session*.

session_type
Defines the SNALINK communication session mode. The *session_type* can have the values of SINGLE, DUAL, or be omitted. If the parameter is omitted, *session_type* defaults to DUAL. If the *session_type* is set to SINGLE, SNALINK creates a single duplex session. If DUAL is specified, SNALINK creates 2 sessions, a send session and a receive session. Like *max_session* and *retry*, if *session_type* is specified, you must also specify the previous parameters.

NCPROUTE and 3745 Communication Controller Ethernet links require *session_type* of SINGLE.

Step 3: Define the SNALINK Application to VTAM

In dual mode, SNALINK opens 2 SNA sessions for each remote logical unit with which it communicates: one for sending and one for receiving. In single mode, SNALINK opens one full-duplex session.

Figure 15 is an example of a typical VTAM APPL statement for SNALINK. The application identifier (SNALKB03 in this example) must match the APPLID specified in the SNALINK cataloged procedure parameters.

```
SNALKB03  APPL ACBNAME=SNALKB03,                                    X
               AUTH=(ACQ,VPACE),                                    X
               SRBEXIT=YES,                                         X
               EAS=12,                                              X
               PARSESS=YES,                                         X
               SONSCIP=YES,                                         X
               VPACING=0
```

Figure 15. APPL Statement for SNALINK

Note: ***SRBEXIT must be yes.***

VTAM Considerations

- Each connection requires 100KB of virtual storage.
- SNALINK provides its own BIND parameters, so it does not assume or require any particular LOGMODE entries.
- The EAS value should be 2 times the number of maximum sessions passed to the SNALINK cataloged procedure.
- SRBEXIT=YES.
- You might have to specify pacing values (VPACING). Consult your VTAM network administrator for further details.
- For *max_ru_size*, be sure to consider the size of the TH, RH, and RU portions. If the maximum size PIU exceeds MAXRU, the NCP will issue a negative response with sense 800A0000 (PIU too long). The definition used in NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. See “HOST Definition Statement” in *VTAM Network Implementation Guide* for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.

Operating SNALINK

If necessary, you can immediately retry a session that is waiting for the retry delay to expire by stopping and starting the SNALINK LU0 interface.

Stopping and Starting SNALINK

To stop SNALINK and close all connections, use the STOP command on the operator’s console. For example, if SNALPROC was the name of the member in the cataloged procedure used to start SNALINK, you would enter:

```
STOP SNALPROC
```

You can also stop SNALINK with the HALT parameter on the MODIFY command. See “Controlling the SNALINK LU0 Interface with the MODIFY Command” on page 342.

SNALINK can be started by:

- Restarting the TCPIP address space if you have included the SNALINK procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

- Issuing START *procname* at the command console (where *procname* is the name of the member in the cataloged procedure used to start the SNALINK LU0 interface.

For example, to restart SNALPROC, enter

```
START SNALPROC
```

Sample Console

The example in Figure 16 and the accompanying information illustrate SNALINK operation.

The line number notations in the example have been added for clarity. They do not appear in the console output.

```
Line 1              | Init complete, APPLID SNALKB03, TCPIP id TCPIPB
Line 2              | Maximum RU size is 00000600
Line 3     SNALKC04 | DLC path 00000001 pending
Line 4     SNALKC04 | Ready to accept bind from remote LU
Line 5     SNALKA04 | DLC path 00000002 pending
Line 6     SNALKA04 | Sending BIND request for SNA send session
Line 7     SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 8     SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 9     SNALKA04 | DLC path 00000002 pending
Line 10    SNALKA04 | Sending BIND request for SNA send session
Line 11    SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 12    SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 13    SNALKC04 | Received BIND request for SNA receive session
Line 14    SNALKC04 | Sending BIND request for SNA send session
Line 15    SNALKC04 | SNA receive session established
Line 16    SNALKC04 | SNA send session established
Line 17    SNALKC04 | Accepting DLC path 00000001
Line 18    SNALKA04 | DLC path 00000002 pending
Line 19    SNALKA04 | Sending BIND request for SNA send session
Line 20    SNALKA04 | SNA send session established
Line 21    SNALKA04 | Accepting DLC path 00000002
Line 22    SNALKA04 | Received BIND request for SNA receive session
Line 23    SNALKA04 | SNA receive session established
Line 24    SNALKC04 | NSEXIT CLEANUP request for receive session
Line 25    SNALKC04 | RECEIVE  CHECK err. R15 00000004 R0 0000000C RTNCD 0000000C FDBK2 0000000B
Line 26    SNALKC04 | RECEIVE  sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 27    SNALKC04 | DLC path 00000001 pending
Line 28    SNALKC04 | Ready to accept bind from remote LU
Line 29    STOP SNALINK
Line 30             | Received STOP command, shutting down
```

Figure 16. SNALINK Console Example

Line Number	Description
Lines 1 and 2	SNALINK displays its startup information from its command line parameters, which are customized as described in “SNALINK Parameters” on page 337. The maximum RU size and all other values are displayed in hexadecimal.
Lines 3 and 4	The TCPIP address space, TCPIPB, issues a DLC CONNECT to establish a session with the remote LU SNALKC04. SNALKC04 is higher in the collating sequence than the local LU name SNALKB03. Consequently, SNALKB03 takes the

	passive role in connecting to SNALKC04, and waits for SNALKC04 to establish a session.
Lines 5 and 6	TCP/IP issues another DLC CONNECT to establish a session with SNALKA04. In this case, SNALKA04 is lower in the collating sequence. Consequently, SNALKB03 takes an active role in connecting to SNALKA04.
Lines 7 and 8	The session establishment attempt to SNALKA04 has failed, as indicated by the (nonzero) return code and the sense information printed.
Lines 9 through 12	Thirty seconds later, TCP/IP again tries to connect to SNALKA04.
Lines 13 and 14	SNALINK receives a BIND request from SNALKC04. SNALINK calls the resulting session the receive session, because it is used only to send data from SNALKC04. Now that the active end has initiated communication, SNALKB03 as the passive end, sends a BIND request to establish a send session.
Lines 15 through 17	The send and receive sessions are fully established. Establishment of the send session causes SNALINK to accept the corresponding DLC path.
Lines 18 through 23	TCP/IP again tries to connect to SNALKA04. This time it is successful (success is indicated by no nonzero return codes).
Lines 24 through 26	SNALKC04 terminates its sessions, and various error messages result.
Lines 27 and 28	Thirty seconds later, TCP/IP again tries to establish communication with SNALKC04. As in lines 13 and 14, SNALKB03 is the passive partner.
Lines 29 and 30	The operator issues a STOP SNALINK command, which causes SNALINK to stop. ***All*** DLC paths and SNA sessions are ended.

Determining the DLC Connection Status Using NETSTAT DEVLINKS

The DLC connect protocol between TCP/IP and SNALINK causes the status of the SNAIUCV device, reported by NETSTAT DEVLINKS, to reflect the status of the SNA sessions to the remote LU.

Status Reported	Explanation
`Issued connect`	Passive side: SNALINK is waiting for a remote LU to establish a session. Active side: SNALINK is trying to establish a session with a remote LU.
`Will retry connect`	The last session was ended, or the last session attempt failed. SNAIUCV driver retries the connection within 30 seconds.

Connected	An SNA send session is established. Under normal conditions this also means a receive session is established or will be established soon, and communication between the two LUs is possible.
Sending message	An SNA send session is established, and there is a DLC SEND currently outstanding.

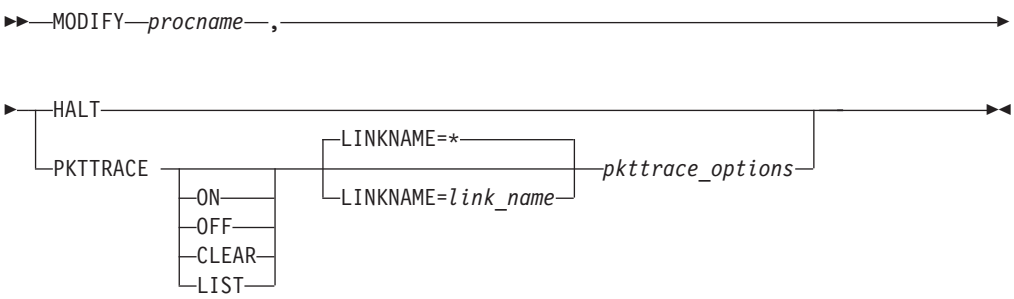
Controlling the SNALINK LU0 Interface with the MODIFY Command

You can stop the SNALINK interface and control SNALINK packet tracing with the MODIFY command. The MODIFY command does not dynamically alter IP to LU mapping.

MODIFY Command—SNALINK LU0

Use the MODIFY command to halt the SNALINK LU0 interface and to enable or disable tracing of IP packets or modify the selection criteria for selecting packets to be traced.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the SNALINK LU0 interface.

HALT
Shuts down the SNALINK interface.

PKTTRACE
Enables or disables tracing of IP packets or modifies the selection criteria for selecting packets to be traced. An IP packet must meet all the conditions specified by the parameters for it to be traced.

If PKTTRACE is specified with no parameters, it is ignored.

ON
Enables packet tracing for the associated link.

OFF
Disables packet tracing for the associated link. The current PKTTRACE settings are retained. Unless these are cleared by a PKTTRACE CLEAR parameter, they will be used by a subsequent PKTTRACE ON parameter.

CLEAR
Disables packet tracing for all links and resets the trace parameters to the default values for the associated link. If the LINKNAME parameter is omitted or an asterisk (*) is specified, tracing is disabled and all trace parameters are set to the default values for all defined links.

LIST
Displays the current state of packet tracing for the associated link. If the LINKNAME parameter is omitted or an (*) is specified, the status of all known links is displayed.

LINKNAME
Specifies the VTAM LU name associated with a SNALINK LU0 DEVICE

statement defined in the TCPIP profile configuration file. If the LINKNAME parameter is omitted or an (*) is specified, the PKTTRACE parameter will apply to all known links.

pkttrace_options
The following *pkttrace_options* are described in “PKTTRACE Statement” on page 223.

- PROT
- IP
- SUBNET
- SRCPORT
- DESTPORT
- FULL
- ABBREV

Examples

Both of the following commands would pass parameters to a SNALINK LU0 address space started with a procedure named SNLK12.TCPSETUP.

```
MODIFY SNLK12.TCPSETUP,HALT

F SNLK12.TCPSETUP,PKTTRACE CLEAR *
```

Usage Notes

TCP/IP for MVS allows the configuration of multiple DLC links to the SNALINK LU0, LU6.2, and X.25 NPSI server address spaces. The PKTTRACE parameter supports this capability through the LINKNAME parameter. Therefore, multiple PKTTRACE parameters can be issued to define the scope of the tracing by identifying the tracing options applicable to multiple links.

PKTTRACE considerations:

- Parsing of the parameter halts as soon as an error is detected and the parameter is ignored.
- Parameters can appear in any order.
- The occurrence of a parameter more than once is an error. In the case of the special parameters ON, OFF, CLEAR, and LIST, the occurrence of more than one of these parameters is an error.
- The PKTTRACE parameter must be issued after the corresponding DLC connection has been accepted from TCPIP.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created or reset using the CLEAR parameter, a link’s trace profile is set to the default values for the trace parameters as follows:
  DESTPORT
  No checking
  FULL Tracing of the whole IP packet
  IP All IP addresses (*)
  PROT All protocols (*)
  SRCPORT
  No checking
  SUBNET
  No checking

- Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (*), which indicates all links. When multiple statements refer to the same link, the parameters on the statements are cumulative, and parameters not specified on the second and subsequent statements are not changed. If a parameter is specified on one statement and then appears on a subsequent statement, the value associated with the last occurrence of the option is used because this is the value that is stored in the trace.

Chapter 9. Configuring the SNALINK LU6.2 Interface

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

The SNALINK LU6.2 cataloged procedure runs a VTAM application program called SNALNK62, which is an interface between the TCPIP address space and the SNA network. SNALNK62 uses SNA LU type 6.2 sessions to pass the TCP/IP data to or from SNALNK62 devices running on other hosts. Examples of SNALNK62 devices include an OS/2 workstation running TCP/IP for OS/2 or a host running TCP/IP for MVS.

This chapter explains how to configure the SNALINK LU6.2 interface and provides information about operating the interface once it is configured. It includes information about starting, stopping, and controlling the SNALINK LU6.2 interface.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

Configuration Process

Steps to configure SNA LU6.2:

1. Specify DEVICE and LINK statements in *hlq*.PROFILE.TCPIP
2. Update the SNALINK LU6.2 cataloged procedure
3. Define the SNALINK LU6.2 application to VTAM
4. Update the SNALINK LU6.2 configuration data set

Step 1: Specify DEVICE and LINK Statements in hlq.PROFILE.TCPIP

You must update the *hlq*.PROFILE.TCPIP data set to include a DEVICE and LINK statement for each DLC connection to be established between the main TCPIP address space and the SNALINK LU6.2 address space. See for details on modifying the *hlq*.PROFILE.TCPIP data set.

Step 2: Update the SNALINK LU6.2 Cataloged Procedure

Update the SNALINK LU6.2 cataloged procedure by copying the sample in *hlq*.SEZAINST(LU62PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. No system parameters are required for the SNALINK LU6.2 address space.

The DD statements in the cataloged procedure should be defined as follows:

DD Name	Description
SYSTCPD	TCPIP.DATA configuration data set
LU62CFG	SNALINK LU6.2 configuration data set
SYSPRINT	Runtime diagnostic or trace output
SYSUDUMP	User abend dump output (optional)

Note: SNALINK LU6.2 does not use the default search sequence to find TCPIP.DATA. Therefore, the //SYSTCPD statement is required.

SNALINK LU6.2 Cataloged Procedure (LU62PROC)

```
//TCPIPL62 PROC MODULE=SNALNK62
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB023
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//SNALNK62 EXEC PGM=&MODULE,TIME=1440,REGION=256K
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//LU62CFG  DD DSN=TCPIP.SEZAINST(LU62CFG),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Step 3: Define the SNALINK LU6.2 Application to VTAM

SNALINK LU6.2 opens 2 SNA LU type 6.2 sessions with each destination node; one for sending and one for receiving. If a destination node supports parallel SNA LU type 6.2 sessions (PARSESS=YES), the 2 sessions use the same remote logical unit; otherwise, 2 remote logical units are used. In either case, SNALINK LU6.2 uses a single local logical unit that must support parallel sessions.

The SNALINK LU6.2 address space must be defined to VTAM as an SNA LU type 6.2 application program. The following APPL statement defines a SNALINK LU6.2 application to VTAM.

Note: ***SRBEXIT must be no.***

```
LU62APPL  APPL ACBNAME=LU62APPL,                                *
               PRTCT=QWERTY,                                    *
               AUTH=(ACQ,VPACE),                                *
               SRBEXIT=NO,                                      *
               EAS=12,                                          *
               PARSESS=YES,                                     *
               SONSCIP=YES,                                     *
               APPC=YES,                                        *
               DLOGMOD=LU62MODE,                                *
               VPACING=0
```

Figure 17. APPL Statement for SNALINK LU6.2

See *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*for further information about defining VTAM applications.

The LOGMODE table entry specified by the APPL DLOGMOD parmeter should have the following form:

```
LU62MODE MODEENT LOGMODE=LU62MODE,FMPROF=X'13',TSPROF=X'07',          *
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',           *
               RUSIZES=X'8585',ENCR=B'0000',                          *
               PSERVIC=X'060200000000000000000300'
```

See *OS/390 SecureWay Communications Server: SNA Customization* for more information about defining log mode tables and *OS/390 eNetwork Communications Server: SNA Programming* for information on PSERVIC values.

Step 4: Update the SNALINK LU6.2 Configuration Data Set

Customize the SNALINK LU6.2 configuration data set by copying the sample provided in *hlq*.SEZAINST(LU62CFG) to your system or recognized PROCLIB and modifying it to suit your local conditions. Add or change the configuration statements as required. Be sure the //LU62CFG statement in the cataloged procedure points to this data set.

Summary of SNALINK LU6.2 Configuration Statements

Table 14. Summary of SNALINK LU6.2 Configuration Statements

Statement	Description	Page
BUFFERS	Specifies the allocation of buffer pools for IP datagrams	351
DEST	Defines and IP-to-LU mapping for a destination node	352
LINK	Defines a link between two TCPIP address spaces	353
TRACE	Defines the required level of tracing for the connections	354
VTAM	Specifies the VTAM application for the connection	355

Sample SNALINK LU6.2 Configuration Data Set (LU62CFG)

```
*--------------------------------------------------------------------*
*   Sample configuration file for SNALNK62                           *
*                                                                    *
*   COPYRIGHT = NONE.                                                *
*                                                                    *
*--------------------------------------------------------------------*
*
*-- VTAM Application definition:
*
*        ApplID    Password
*        --------  --------
VTAM     LU62APPL  QWERTY
*
*-- Link Definition:
*                                    Idle
*        TCP/IP Link Name  LogMode   Time-out
*        ----------------  --------  --------
LINK     LNKLU62           LU62MODE  600
*
*-- Destination address list for this link:
*
*                         *--- REMOTE ----*  Start
*        IP Address       Send LU  Recv LU   Type
*        ---------------  -------- --------  ----
DEST     192.9.207.39     LU6LBK11 LU6LBK12  INIT
DEST     192.9.207.40     LU6LBK13 =
```

```
DEST     192.9.207.41     LU6LBK14 =
*
*-- Buffer specifications:
*
*        Datagram  Add Snd  Snd Queue
*        Max Size  Buffers  Limit
*        -----     -----    -----
BUFFERS  32758     10       11
*
*-- Trace level specifications:
*
*      Trace
*      Level   Connection Range
*      ------  ------------------
TRACE  OFF     ALL
```

SNALINK LU6.2 Configuration Statements

This section contains the statements for the SNALINK LU6.2 configuration data set.

Statement Syntax

Use the following syntax rules:

- Each statement must be entered on a separate line in the configuration data set.
- Each statement consists of a keyword, followed by one or more parameter fields separated by a blank.
- Case is not significant and leading blanks are ignored.
- Comment lines are marked with an asterisk in column 1.

Statement Ordering

- At least one of each type of statement, except the TRACE statement, must appear in the configuration data set. TRACE is the only optional statement.
- Only one BUFFERS statement can be defined, and it can appear anywhere in the data set.
- Only one VTAM statement can be defined, and it must appear before the first LINK statement.
- You must have one LINK statement for each network ***directly connected*** to the SNALINK LU6.2 address space.
- You must have one DEST statement for each distinct IP address on the directly connected networks.
- A DEST statement defining a destination IP address must appear before a TRACE statement for the same destination IP address. The DEST statements for a particular network must appear after the LINK statement for that network and before the next LINK statement.

  For example:

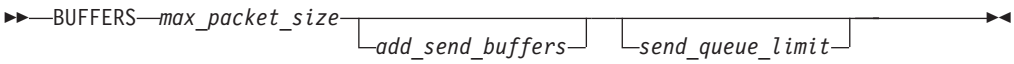
  ```
  LINK
  DEST
  .
  .
  .
  LINK
  ```

- The data set can have any number of TRACE statements. If the ranges specified in any of the TRACE statements overlap, the resulting trace levels are determined by invoking the TRACE statements in the order in which they appear in the data set.

BUFFERS Statement

Use the BUFFERS statement to specify the parameters used to allocate buffer pools for storing IP datagrams.

Syntax



Parameters

max_packet_size
The maximum IP packet size. This value should match the *max_packet_size* parameter on the GATEWAY statement in the *hlq*.PROFILE.TCPIP data set and it must be less than the maximum physical information unit (PIU) in the VTAM definition for your LU6.2 connection. The maximum PIU is set by the MAXDATA parameter of the PCCU macro which is part of the NCP generation program in the VTAMLST library. The value for this parameter differs for each device. Check the documentation for your device to determine the appropriate value.

The *max_packet_size* value must be an integer in the range 20 to 32758. Any datagrams exceeding this length that are received from either the local or remote TCP/IP are discarded.

add_send_buffers
The number of additional buffers for storing datagrams waiting to be passed to VTAM. These are in addition to the initial allocation of one for every destination (DEST) defined in the SNALINK LU6.2 configuration data set. If an *add_send_buffers* value is not specified, no additional buffers are allocated. The minimum value allowed for *add_send_buffers* is 1; the maximum value allowed is 2147483647.

send_queue_limit
The maximum number of buffers that can be allocated to any connection for storing datagrams waiting to be passed to VTAM. This parameter allows you to prevent a single connection from restricting other connections access to the free buffers from the VTAM send buffer pool. Once this maximum limit is reached on the VTAM send queue, further datagrams received from TCP/IP for this connection are discarded, until the number of buffers on the send queue is reduced below the limit. For this parameter to be effective, its value should leave enough free buffers in the VTAM send buffer pool to service the other active connections when full throughput is reached on the main connections. See the description of the *add_send_buffers* parameter to calculate the size of the VTAM send buffer pool. If a *send_queue_limit* value is not specified, no limit is placed on the lengths of individual send queues. The minimum value allowed for *send_queue_limit* is 1; the maximum value allowed is 2147483647.

Examples

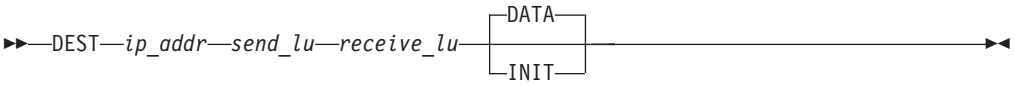
```
*-- Buffer specifications:
*
*        Datagram  Add Snd  Snd Queue
*        Max Size  Buffers  Limit
*        -----     -----    -----
BUFFERS  32758     10       11
*
```

DEST Statement

Use the DEST statement to define an IP-address-to-LU-name mapping for a destination node associated with the link specified in the most recent LINK statement.

The IP addresses listed must be consistent with the direct connections defined for the current link in the GATEWAY statement of the *hlq*.PROFILE.TCPIP data set.

Syntax



Parameters

ip_addr
The IP address in dotted decimal format. The value entered must be in the correct format for an IP address of a network node (a fully qualified IP address).

send_lu
The remote LU name for the ***send*** connection.

receive_lu
The remote LU name for the ***receive*** connection. For independent logical units using parallel sessions, the send and receive LU names are the same. In this case, you can enter an equal sign (=) as the value for the *receive_lu* parameter.

DATA
Definition of when the connection to the destination node is to be established. If the DATA parameter is specified, the connection is only established after there is IP data to be transferred to and from the destination node. If neither INIT nor DATA is specified, DATA is used as the default setting.

INIT
Definition of when the connection to the destination node is to be established. If the INIT parameter is specified, the connection is established during initialization of the SNALINK LU6.2 address space.

Examples

```
*        IP Address       Send LU  Recv LU   Type
*        ---------------  -------- --------  ----
DEST     192.9.207.39     LU6LBK11 LU6LBK12  INIT
DEST     192.9.207.40     LU6LBK13 =
DEST     192.9.207.41     LU6LBK14 =
```

LINK Statement

Use the LINK statement to define the link between the main TCPIP address space and the SNALINK LU6.2 interface.

Syntax

Parameters

link_name
The name of the TCP/IP link, as defined in the *hlq*.PROFILE.TCPIP data set to which the destinations in the subsequent DEST statements are to apply. The maximum length is 8 characters.

log_mode
The name of the VTAM LOGMODE entry to be used when establishing an SNA LU type 6.2 session for this link.

idle_disconnect
Time, in seconds, after which an idle or inactive session is terminated. If blank or zero, no inactivity checking or time-out is to apply to connections defined for this link. If not blank, the value of this parameter must lie within the inclusive range 0 to $2^{31}-1$.

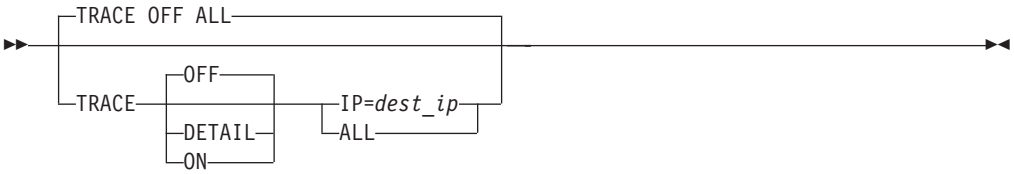
Examples

```
*-- Link Definition:
*                                   Idle
*        TCP/IP Link Name  LogMode   Time-out
*        ----------------  --------  --------
LINK     LNKLU62           LU62MODE  600
```

TRACE Statement

Use the TRACE statement to define the required level of tracing for a specified range of connections. All tracing levels default to OFF unless overridden by any appropriate TRACE statement. Trace output is written to the SYSPRINT data set.

Syntax



Parameters

OFF
Disables tracing for all connections in the specified range. If OFF, DETAIL, or ON is not specified, OFF is the default. For example, specifying `TRACE IP=dest_ip` would disable tracing for only the connection associated with *dest_ip*.

DETAIL
Enables a detailed level of tracing for all connections in the specified range.

ON
Enables a basic level of tracing for all connections in the specified range.

IP=*dest_ip*
The destination IP address associated with the connection for which tracing is enabled or disabled. A DEST statement defining a destination IP address must appear before a TRACE statement for the same destination IP address.

ALL
If the ALL parameter is specified, tracing for all destinations (either currently defined or still to be defined) is set to the requested level.

Examples

```
*-- Trace level specifications:
*
*      Trace
*      Level   Connection Range
*      ------  ------------------
TRACE  OFF     ALL
```

VTAM Statement

Use the VTAM statement to specify the VTAM application definition to be used to establish the connections.

The VTAM statement must precede the first LINK statement.

Syntax

Parameters

application_id
The VTAM application identifier as defined by the VTAM APPL statement. This name identifies the local logical unit used by this SNALINK LU6.2 address space. Remote SNALINK LU6.2 interfaces configure this name as their remote logical unit name for a connection to this SNALINK LU6.2 address space.

The maximum length is 8 characters.

password
The password for the VTAM application specified in *application_id*. This must match the password specifed by the PRCT parameter in the VTAM APPL statement.

Examples

The following example shows the connection between the VTAM APPL statement and the VTAM configuration statement.

```
LU62APPL  APPL ACBNAME=LU62APPL,                                  *
               PRTCT=QWERTY,                                      *

*-- VTAM Application definition:
*
*        ApplID    Password
*        --------  --------
VTAM     LU62APPL  QWERTY
```

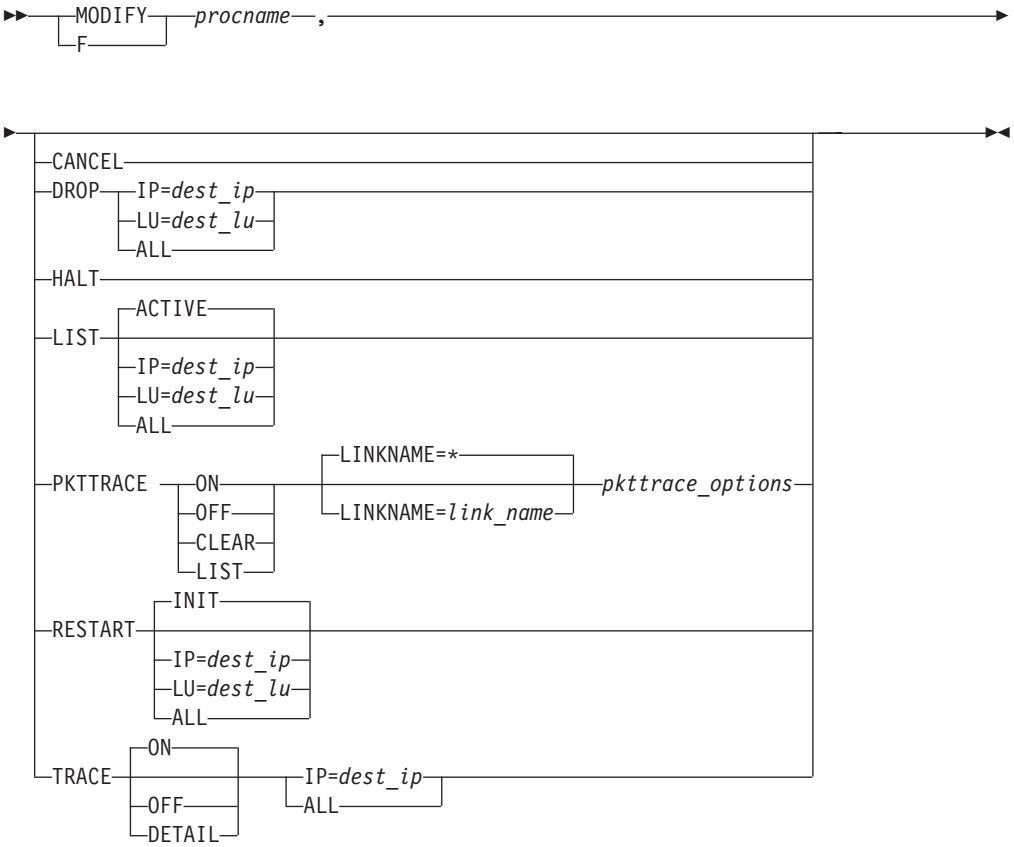
Controlling the SNALINK LU6.2 Interface: This section contains information for controlling the SNALINK LU6.2 interface.

MODIFY Command—SNALINK LU 6.2

You can stop or restart the SNALINK LU6.2 interface and control tracing with the MODIFY command. Use the MODIFY command to:

- Stop or restart the SNALINK LU6.2 interface
- Enable or disable tracing of IP packets
- Modify the selection criteria for selecting packets to be traced

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the SNALINK LU6.2 interface.

CANCEL
Cancels the SNALINK LU6.2 interface by a user abend. The system produces a dump and writes it to the data set defined by the //SYSUDUMP DD statement in the cataloged procedure.

DROP
Ends the connection with the destination nodes as specified.

IP=*dest_ip*
The destination IP address of the connection to end.

LU=*dest_lu*
The destination LU name of the connection to end. For dependent LU

connections, either the sending or receiving remote LU name can be supplied and both sessions are ended.

ALL Drops all connections defined in SNALINK LU6.2 configuration data set.

HALT
Shuts down the SNALINK LU6.2 interface.

LIST
Displays status and traffic information for the range of connections specified.

ACTIVE
The range of destinations to be listed. Information is displayed for all currently established connections handled by the specified address space. This is the default.

IP=*dest_ip*
The destination IP address of the connection to be listed.

LU=*dest_lu*
The destination LU name of the connection to be listed. For dependent LU connections, you can supply either the remote sending or receiving LU name.

ALL Displays information for all destinations defined in the SNALINK LU6.2 configuration data set.

PKTTRACE
Enables or disables tracing of IP packets or modifies the selection criteria for selecting packets to be traced. An IP packet must meet all the conditions specified by the parameters for it to be traced.

If PKTTRACE is specified with no parameters, it is ignored.

ON
Enables packet tracing for the associated link.

OFF
Disables packet tracing for the associated link. The current PKTTRACE settings are retained. Unless these are cleared by a PKTTRACE CLEAR parameter, they will be used by a subsequent PKTTRACE ON parameter.

CLEAR
Disables packet tracing for all links and resets the trace parameters to the default values for the associated link. If the LINKNAME parameter is omitted or an asterisk (*) is specified, tracing is disabled and all trace parameters are set to the default values for all defined links.

LIST
Displays the current state of packet tracing for the associated link. If the LINKNAME parameter is omitted or an (*) is specified, the status of all known links is displayed.

LINKNAME
Specifies the VTAM LU name associated with a SNALINK LU6.2 DEVICE statement defined in the TCPIP profile configuration file. If the LINKNAME parameter is omitted or an (*) is specified, the PKTTRACE parameter will apply to all known links.

pkttrace_options
The following *pkttrace_options* are described in “PKTTRACE Statement” on page 223.

- PROT
- IP
- SUBNET
- SRCPORT
- DESTPORT
- FULL
- ABBREV

RESTART
Establishes one or more connections to destination nodes. Any destinations in the specified range that are already connected are skipped.

INIT The range of connections to be established. If the INIT parameter is specified, connections are established with all destinations defined with the INIT parameter in the SNALINK LU6.2 configuration data set. If the RESTART subcommand is entered without parameters, the INIT option is the default.

IP=*dest_ip*
The destination IP address of the connection to be established.

LU=*dest_lu*
The destination LU name of the connection to be established. For dependent LU connections, either the remote sending or receiving LU name can be supplied and both sessions are established.

ALL The range of connections to be established. If the ALL parameter is specified, connections are established with all destinations defined in the SNALINK LU6.2 configuration data set.

TRACE
Alters the levels of trace defined in the SNALINK LU6.2 configuration data set while the address space is active.

ON Enables a basic level of tracing for all connection in the specified range. The default is ON.

OFF If the OFF parameter is specified, tracing is disabled for all connections in the specified range.

DETAIL
Enables a detailed level of tracing for all connections in the specified range.

IP=*dest_ip*
The destination IP address associated with the connection for which tracing will be enabled or disabled.

ALL If the ALL parameter is specified, tracing for all destinations (either currently or subsequently connected) is set to the requested level.

Examples

To enable tracing for the procedure LU62PROD on connection associated with 9.163.37.12, enter

```
F LU62PROD, TRACE IP=9.163.37.112
```

The following example illustrates the output you might get if you issued the MODIFY command with the LIST parameter:

```
MODIFY TCPIPL62,LIST ALL

    TCPL62217I LIST Accepted; Range = All Connections
```

```
TCPL62212I   192.9.207.39 (Connected on 92.013 at 09:52:11)
TCPL62213I     Connected by:  DATA             Trace Level: OFF
TCPL62214I     SEND:-  Status: Not Allocated    Packets Out: 0
TCPL62215I     RECV:-  Status: Allocated        Packets In:  0
TCPL62211I   192.9.207.40 (Disconnected on 92.013 at 08:30:10)
TCPL62210I   192.9.207.41 (Disconnected)
TCPL62219I LIST Completed
```

Usage Notes

TCP/IP for MVS allows the configuration of multiple DLC links to the SNALINK LU0, LU6.2, and X.25 NPSI server address spaces. The PKTTRACE parameter supports this capability through the LINKNAME parameter. Therefore, multiple PKTTRACE parameters can be issued to define the scope of the tracing by identifying the tracing options applicable to multiple links.

PKTTRACE considerations:

- Parsing of the parameter halts as soon as an error is detected and the parameter is ignored.
- Parameters can appear in any order.
- The occurrence of a parameter more than once is an error. In the case of the special parameters ON, OFF, CLEAR, and LIST, the occurrence of more than one of these parameters is an error.
- The PKTTRACE parameter must be issued after the corresponding DLC connection has been accepted from TCPIP.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created, or reset using the CLEAR parameter, a link's trace profile is set to the default values for the trace parameters as follows:

  DESTPORT No checking

  FULL Tracing of the whole IP packet

  IP All IP addresses (*)

  PROT All protocols (*)

  SRCPORT No checking

  SUBNET No checking
- Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (*), which indicates all links. When multiple statements refer to the same link, the parameters on the statements are cumulative, and parameters not specified on the second and subsequent statements are not changed. If a parameter is specified on one statement and then appears on a subsequent statement, the value associated with the last occurrence of the option is used because this is the value that is stored in the trace.

Determining the DLC Connection Status Using NETSTAT DEVLINKS: For the SNALINK LU6.2 interface, the connection and disconnection of DLC links between the TCPIP and SNALINK LU6.2 address spaces is independent of the connection and disconnection of VTAM links with destination nodes.

You can use the TSO command, NETSTAT DEVLINKS, to determine the status of the DLC connections between the main TCPIP address space and the SNALINK LU6.2 address spaces.

Status Reported	Description
Inactive	The DLC connection has not been started. You can start one of the DLC links between TCP/IP and SNALINK LU6.2 with the VARY START command.
Issued Connect	The TCPIP address space has issued a DLC connection request, but the SNALINK LU6.2 address space has not yet accepted the connection.
Connected	A DLC connection has been successfully established between the TCPIP address space and the SNALINK LU6.2 address space.
Sending Message	A DLC connection has been successfully established between the 2 address spaces, and a message has been sent by the TCPIP address space, but it has not yet been received by the SNALINK LU6.2 address space.
Will retry connect	Either a previously connected DLC connection has been severed, or the previous connection request was not accepted within the time-out period. In either case, the TCPIP address space attempts to resend another connection request within 30 seconds.

Stopping and Starting SNALINK LU6.2 Interface: To stop the SNALINK LU6.2 interface and close all connections, use the STOP command on the operator's console. For example, if LU62PROC was the name of the member in the cataloged procedure used to start SNALINK LU6.2, you would enter:

```
STOP LU62PROC
```

You can also stop SNALINK LU6.2 with the HALT or CANCEL parameters on the MODIFY command. See "MODIFY Command—SNALINK LU 6.2" on page 356.

SNALINK LU6.2 can be started by:

- Restarting the TCPIP address space if you have included the SNALINK LU6.2 procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set
- Issuing START *procname* at the command console (where *procname* is the name of the member in the cataloged procedure used to start the SNALINK LU6.2 interface.

For example, to restart LU62PROC, enter

```
START LU62PROC
```

Chapter 10. Configuring the X.25 NCP Packet Switching Interface (NPSI) Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

The X.25 NPSI server runs a VTAM application program called XNX25IPI, which is the interface between the TCPIP address space’s DLC driver and your X.25 network. XNX25IPI communicates with the X.25 NCP Packet Switching Interface in a front-end IBM 37xx Communications Controller.

Large scale X.25 network applications often require multiple physical lines to the network switch for increased capacity and reliability. You can configure the X.25 NPSI server to support multiple lines as a group, rather than individually. In this configuration, the collection of lines are assigned a single address called a ***hunt group*** address. Incoming X.25 calls are distributed among the lines in either rotary or traffic balancing fashion, depending on the services offered by the X.25 network provider.

This chapter explains how to configure the X.25 NPSI server and provides information about using the MODIFY command to operate the server once it is running. For information on improving the performance of the X.25 NPSI network, see the options on the PORT statement (page 229) and GATEWAY statement (page 193) in the *hlq*.PROFILE.TCPIP and the explanation provided in the *TCP/IP: Performance Tuning Guide.*

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

Configuration Process

This section describes how to configure the X.25 NPSI server.

Steps to Configure the X.25 NPSI Server:

1. Specify X.25 configuration statements in *hlq*.PROFILE.TCPIP
2. Update the X.25 NPSI cataloged procedure
3. Modify the X.25 NPSI server configuration data set
4. Define the X.25 NPSI configuration
5. Define the X.25 NPSI application to VTAM
6. Define VTAM Switched Circuits

If you want to run the X.25 NPSI cataloged procedure in a different domain than the X.25 NPSI communication controller, see “Configuring Cross-Domain Resources” on page 370.

For information about operating the X.25 NPSI server with the MODIFY command, see “Operating the X.25 NPSI Server Using the MODIFY Command” on page 382.

Step 1: Specify X.25 Configuration Statements in hlq.PROFILE.TCPIP

To configure the *hlq*.PROFILE.TCPIP data set for X.25 NPSI, include appropriate DEVICE, LINK, HOME, GATEWAY, and START statements. The following example shows the statements that would correspond with the other X.25 samples in this chapter.

```
;
DEVICE X25DEV X25NPSI TCPIPX25
LINK X25LINK SAMEHOST 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; Network  First hop   Link name Packet size  Subnet mask  Subnet value
  192.005      =        X25LINK     2000       0.0.255.0    0.0.58.0
;
START X25DEV
;
```

Note: Only one DEVICE and LINK statement per TCPIPX25 address space is allowed.

Step 2: Update the X.25 NPSI Cataloged Procedure

Update the X.25 NPSI cataloged procedure by copying the sample provided in *hlq*.SEZAINST(X25PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions.

Change the data set names as needed:

- The X.25 interface does not use the TCP/IP default search sequence for *hlq*.TCPIP.DATA, therefore //SYSTCPD DD is required.
- Modify the //X25IPI DD statement to point to your X.25 configuration data set.

X.25 NPSI Cataloged Procedure (X25PROC):

```
//TCPIPX25 PROC MODULE=XNX25IPI
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEB020
//*
//*        5655-HAL (C) Copyright IBM Corp. 1989, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//*
//X25IPI   EXEC PGM=&MODULE,REGION=256K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//X25IPI   DD DSN=TCPIP.SEZAINST(X25CONF),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
```

```
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 3: Modify the X.25 NPSI Server Configuration Data Set

A sample configuration data set provided in *hlq*.SEZAINST(X25CONF) gives examples of how to define a public network connection, a Defense Data Network connection, and private point-to-point connection to a router. Copy this sample to the data set pointed to by the //X25IPI DD statement in your X.25 NPSI cataloged procedure. Update this sample to define your X.25 connections using the statements listed in Table 15.

Each connection must have a LINK and at least one DEST statement. You can optionally define hunt groups, fast connects, and call handling options for each link, and global options such as trace levels, when to clear inactive connections, and the buffer size to use for IP datagrams. You can find complete syntax for each of these statements in “X.25 NPSI Server Configuration Statements” on page 370.

Summary of X.25 NPSI Server Configuration Statements

Table 15. Summary of X.25 NPSI Server Configuration Statements

Statement	Description	Page
ALTLINK	Specifies the members of a link’s hunt group for incoming calls	371
BUFFERS	Specifies the buffer size to use for IP datagrams	373
DEST	Specifies the destination address list for a link	374
FAST	Specifies that a link will have NPSI GATE fast connect	375
LINK	Defines the link to a NPSI physical circuit logical unit	376
OPTIONS	Specifies the call handling options for incoming calls on a link	377
TIMERS	Defines the time limits for holding or clearing connections on all links	379
TRACE	Specifies the trace and debug levels for the X.25 NPSI server	380
VTAM	Identifies the VTAM APPL definition for the X.25 NPSI server	382

Sample X.25 NPSI Server Configuration Data Set (X25CONF)

```
*--------------------------------------------------------------------*
*   Sample configuration file for TCPIPX25                           *
*                                                                    *
*   COPYRIGHT = NONE.                                                *
*--------------------------------------------------------------------*
*-- Trace level and debug flags
*                  01234567
Trace    OFF       00000000
*
*-- VTAM Application definition:
*        ApplID    Password
*        --------  --------
VTAM     TCPIPX25  TCPX25
*
*--------------------------------------------------------------------*
*   Definitions for a public network connection with two-line hunt
*   group, using NPSI GATE Fast Connect. Network default packet size
*   is 128; packet size 1024 negotiated on call request
*
```

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU021     3110 23456789        2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890        2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
*
*   Destination address list for this link            Destination
*        IP Address      X.25 DTE Addr       C.U.D.   Facilities
*        --------------- ---------------     -------- ---------------
Dest     192.005.058.001 131106170015300     CC
Dest     192.005.058.004 131106170015320     CC
Dest     192.005.058.005 131106170015350     CC       02AA
*                            * this dest. requires throughput class on
*                                          call request
*
*--------------------------------------------------------------------*
*   Definitions for a DDN connection
*     Note: DDN and non-DDN links cannot be mixed
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
*Link     XU023     DDN                 2        128   16
*Options  GATE
*
*   DDN network 10 uses RFC 1236 address calculation, no explicit
*   X.25 DTE addresses
*
*        IP Address      X.25 DTE Addr
*        --------------- ---------------
*Dest     10
*
*--------------------------------------------------------------------*
*   Definitions for a private point-to-point link to a router
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU024     PRIV 1               2      1024   2
Options  GATE
*
*        IP Address      X.25 DTE Addr       C.U.D.
*        --------------- ---------------     --------
Dest     192.5.57.2      2
*
*--------------------------------------------------------------------*
*-- Buffer specifications:
*        Datagram  Addn'l    VC Queue
*        Max Size  Buffers   Limit
*        --------  --        --
Buffers  1024      50        4
*
*-- Timers:
*        Idle      Minimum
*        Disconn.  Open
*        --------  -------
Timers   300       60
```

Step 4: Define the X.25 NPSI Configuration

Define the X.25 NPSI configuration according to the information in *X.25 NPSI Planning and Installation* The X.25 NPSI server supports use of the LOGAPPL operand on the X25.MCH definition in the X.25 NPSI configuration to allow automatic recovery. You can use either the Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE).

IBM recommends using the X.25 NPSI GATE configuration which allows sharing of an X.25 physical link and provides better error recovery. A sample is provided in *hlq*.SEZAINST(NPSIGATE). NPSI GATE requires that you include the OPTIONS GATE statement in the X.25 NPSI configuration data set after the LINK statement, as shown in this portion of the X25CONF sample:

```
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU024     PRIV 1               2      1024   2
Options  GATE
*
*        IP address      X.25 DTE addr       C.U.D.
*        --------------- ---------------     --------
Dest     192.5.57.2      2
```

Sites that need to use the X.25 NPSI DATE configuration can find a sample in inside of PV *hlq*.SEZAINST(NPSIDATE). See *X.25 NPSI Host Programming* for information about the definitions and parameters used in these configurations.

This shows portions of the sample NPSI GATE configuration (NPSIGATE). Ellipses (....) indicate code that has been omitted.

```
 ***********************************************************************
          OPTIONS NEWDEFN=YES,USERGEN=X25NPSI
 ***********************************************************************
.....
.....
 NPSIV32  BUILD ADDSESS=400,                                          +
                AUXADDR=800,                                          +
                ERLIMIT=16,                                           +
                NAMTAB=120,                                           +
                MAXSESS=250,                                          +
                USGTIER=5,                                            +
                BRANCH=8000,                                          +
                BFRS=104,           BUFFER SIZE TO BE GENED           +
                CATRACE=(YES,255),  CHAN.ADAPTER TRACE OPTION         +
                CSMSG=C3D9C9E340E2C9E340D4C5E2E2C1C7C540C6D6D940E2E24040+
                40C2C340E3C5D9D4C9D5C1D3,                             +
                CWALL=26,                                             +
                ENABLTO=30.0,                                         +
                ERASE=YES,                                            +
                LOADLIB=NCPLOAD,    TARGET OF FINAL LINKEDIT          +
                LTRACE=8,           LINES TRACED SIMULTANEOUSLY       +
                MAXSSCP=8,          NUMBER OF CONCURRENT SSCP'S       +
                MODEL=3745,                                           +
                VERSION=V5R2.1,                                       +
                NEWNAME=NPSITCP,    NAME OF NCP LOAD MODULE           +
                NUMHSAS=8,          HOST SA IN CONCURRENT COMMUNICATION +
                OLT=YES,            ONLINE TERMINAL TEST              +
                PWROFF=YES,                                           +
                BACKUP=500,                                           +
                SALIMIT=511,                                          +
                SLODOWN=12,         BUFFER SLOWDOWN THRESHOLD (PERCENT) +
```

```
               SUBAREA=03,                                              +
               TRACE=(YES,100),    ADDRESS TRACE OPTION IN CORE TABLE   +
               TYPGEN=NCP,                                              +
               TYPSYS=MVS,         NCP TO BE GENERATED ON MVS           +
               TWXID=(E8D6E4C3C1D3D311,C2C9C7D5C3D7C3C1D3D325),         +
               VRPOOL=30,                                               +
               TRANSFR=32,                                              +
               NETID=NETA,                                              +
               X25.USGTIER=5,                                           +
               X25.IDNUMH=01,                                           +
               X25.MCHCNT=4,                                            +
               X25.MAXPIU=64K
.....
.....
*************************************************************************
*
*                  NPSI DEFINITIONS
*
*************************************************************************
X25XXX   X25.NET CPHINDX=1,                                             +
               NETTYPE=1,                                               +
               DM=YES,                                                  +
               OUHINDX=1
         X25.VCCPT INDEX=1,                                             +
               MAXPKTL=128,                                             +
               VWINDOW=2
         X25.OUFT INDEX=1
*-----------------------------------------------------------------------*
*        Hunt group primary line 021 with fast connect                  *
*-----------------------------------------------------------------------*
HGRP01A  X25.MCH ADDRESS=21,                                            +
               FRMLGTH=131,        128 byte packet + 3 byte header      +
               PKTMODL=8,                                               +
               ANS=CONT,                                                +
               LCGDEF=(0,16),      16 logical channels in group 0       +
               MWINDOW=2,                                               +
               STATION=DTE,                                             +
               SPEED=9600,                                              +
               LCN0=NOTUSED,                                            +
               GATE=GENERAL,       GATE                                 +
               LLCLIST=(LLC4),                                          +
               CONNECT=YES,        Fast connect                         +
               LOGAPPL=TCPIPX25,                                        +
               DBIT=NO,                                                 +
               DIRECT=NO,                                               +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                             +
               MAXDATA=1034,       MAXDATA only with Fast connect!      +
               TYPE=SWITCHED,                                           +
               CALL=INOUT,                                              +
               OUFINDX=1,                                               +
               VCCINDX=1
*-----------------------------------------------------------------------*
*        Hunt group secondary line 022 with fast connect                *
*-----------------------------------------------------------------------*
HGRP01B  X25.MCH ADDRESS=22,                                            +
               FRMLGTH=131,        128 byte packet + 3 byte header      +
               PKTMODL=8,                                               +
               ANS=CONT,                                                +
               LCGDEF=(0,16),      16 logical channels in group 0       +
               MWINDOW=2,                                               +
               STATION=DTE,                                             +
               SPEED=9600,                                              +
               LCN0=NOTUSED,                                            +
               GATE=GENERAL,       GATE                                 +
               LLCLIST=(LLC4),                                          +
```

```
               CONNECT=YES,        Fast connect                        +
               LOGAPPL=TCPIPX25,                                       +
               DBIT=NO,                                                +
               DIRECT=NO,                                              +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                            +
               MAXDATA=1034,       MAXDATA only with Fast connect!     +
               TYPE=SWITCHED,                                          +
               CALL=INOUT,                                             +
               OUFINDX=1,                                              +
               VCCINDX=1
*-------------------------------------------------------------------------*
*        DDN line 023                                                    *
*-------------------------------------------------------------------------*
DTE01    X25.MCH ADDRESS=23,                                           +
               FRMLGTH=131,        128 byte packet + 3 byte header     +
               PKTMODL=8,                                              +
               ANS=CONT,                                               +
               LCGDEF=(0,16),      16 logical channels in group 0      +
               MWINDOW=2,                                              +
               STATION=DTE,                                            +
               SPEED=9600,                                             +
               LCN0=NOTUSED,                                           +
               GATE=GENERAL,       GATE                                +
               LLCLIST=(LLC4),                                         +
               LOGAPPL=TCPIPX25,                                       +
               CTCP=(00),          paired with CUD list                +
               CUD0=(CC),          incoming CUD selects CTCP           +
               DBIT=NO,                                                +
               DIRECT=NO,                                              +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                            +
               TYPE=SWITCHED,                                          +
               CALL=INOUT,                                             +
               OUFINDX=1,                                              +
               VCCINDX=1
*-------------------------------------------------------------------------*
*        Private line 024: DCE station to router                        *
*-------------------------------------------------------------------------*
DCE01    X25.MCH ADDRESS=24,       1024 byte packet + 3 byte header    +
               FRMLGTH=1027,                                           +
               PKTMODL=8,                                              +
               ANS=CONT,                                               +
               LCGDEF=(0,2),                                           +
               MWINDOW=2,                                              +
               STATION=DCE,                                            +
               SPEED=9600,                                             +
               LCN0=NOTUSED,                                           +
               GATE=GENERAL,                                           +
               LLCLIST=(LLC4),                                         +
               CTCP=(00),          paired with CUD list                +
               CUD0=(CC),          incoming CUD selects CTCP           +
               DBIT=NO,                                                +
               DIRECT=NO,                                              +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,2),                                             +
               TYPE=SWITCHED,                                          +
               CALL=INOUT,                                             +
               OUFINDX=1,                                              +
               VCCINDX=1
         X25.END
```

```
 ********************************************************************
....
....
 GENEND   GENEND
```

Step 5: Define the X.25 NPSI Application to VTAM

Define the X.25 NPSI VTAM application with an APPL statement in the VTAMLST. Following is an example of a VTAM APPL statement for X.25 NPSI.

```
          VBUILD TYPE=APPL                                          +
 TCPIPX25 APPL  ACBNAME=TCPIPX25,                                   +
                PRTCT=TCPX25,                                       +
                AUTH=(ACQ),                                         +
                PARSESS=YES,                                        +
                EAS=20
```

Step 6: Define VTAM Switched Circuits

X.25 NPSI switched virtual circuits (SVCs) appear to VTAM as switched links, [2] requiring a switched circuit definition of a physical unit (PU) and logical unit (LU) for each SVC. The sample provided in *hlq*.SEZAINST(X25VSVC) shows the definitions of a VTAM switched circuit corresponding to the sample X.25 NPSI GATE configuration.

The definitions are associated with the SVCs by identifying numbers (IDNUMs) created automatically during X.25 NPSI generation. The entries, in hexadecimal, run in steps of 2, by default, in the opposite order to the MCH and SVC definitions in the X.25 NPSI configuration.

Notes:

1. Permanent virtual circuits (PVCs) are not supported.
2. If you specify a local version of the USS table with the SSCPFM operand, it must not have an entry for message 10 (the welcome message); otherwise, the X.25 NPSI server does not operate correctly.

This is the sample SVC configuration data set (X25VSVC):

```
          VBUILD TYPE=SWNET,MAXGRP=1,MAXNO=1
 *--------------------------------------------------------------------*
 *   Switched circuits for DDN line 023 (16 VCs, IDNUMS 006-024)      *
 *                                                                    *
 *   COPYRIGHT = NONE.                                                *
 *--------------------------------------------------------------------*
 VP023001 PU    ADDR=23,IDBLK=003,IDNUM=01024,                       +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,      +
                SSCPFM=USSNTO
 VL023001 LU    LOCADDR=0
 VP023002 PU    ADDR=23,IDBLK=003,IDNUM=01022,                       +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,      +
                SSCPFM=USSNTO
 VL023002 LU    LOCADDR=0
 VP023003 PU    ADDR=23,IDBLK=003,IDNUM=01020,                       +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,      +
                SSCPFM=USSNTO
 VL023003 LU    LOCADDR=0
 VP023004 PU    ADDR=23,IDBLK=003,IDNUM=0101E,                       +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,      +
```

2. except when using fast connect, where they appear as leased lines to VTAM. For more information, see “FAST Statement” on page 375.

```
                SSCPFM=USSNTO
VL023004 LU     LOCADDR=0
VP023005 PU     ADDR=23,IDBLK=003,IDNUM=0101C,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023005 LU     LOCADDR=0
VP023006 PU     ADDR=23,IDBLK=003,IDNUM=0101A,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023006 LU     LOCADDR=0
VP023007 PU     ADDR=23,IDBLK=003,IDNUM=01018,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023007 LU     LOCADDR=0
VP023008 PU     ADDR=23,IDBLK=003,IDNUM=01016,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023008 LU     LOCADDR=0
VP023009 PU     ADDR=23,IDBLK=003,IDNUM=01014,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023009 LU     LOCADDR=0
VP023010 PU     ADDR=23,IDBLK=003,IDNUM=01012,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023010 LU     LOCADDR=0
VP023011 PU     ADDR=23,IDBLK=003,IDNUM=01010,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023011 LU     LOCADDR=0
VP023012 PU     ADDR=23,IDBLK=003,IDNUM=0100E,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023012 LU     LOCADDR=0
VP023013 PU     ADDR=23,IDBLK=003,IDNUM=0100C,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023013 LU     LOCADDR=0
VP023014 PU     ADDR=23,IDBLK=003,IDNUM=0100A,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023014 LU     LOCADDR=0
VP023015 PU     ADDR=23,IDBLK=003,IDNUM=01008,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023015 LU     LOCADDR=0
VP023016 PU     ADDR=23,IDBLK=003,IDNUM=01006,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL023016 LU     LOCADDR=0
*-------------------------------------------------------------------*
*   Switched circuits for private line 024 (2 VCs, IDNUMS 002-004)  *
*-------------------------------------------------------------------*
VP024001 PU     ADDR=24,IDBLK=003,IDNUM=01004,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL024001 LU     LOCADDR=0
VP024002 PU     ADDR=24,IDBLK=003,IDNUM=01002,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
VL024002 LU     LOCADDR=0
```

Configuring Cross-Domain Resources

If you run the TCP/IP X.25 NPSI server in a different domain from the communication controller running X.25 NPSI, place the following X.25 definitions in the following manner:

- Define the X.25 NPSI server in the domain under which it runs and as a cross-domain resource in the X.25 NPSI domain.
- Place cross-domain resource definitions for the X.25 NPSI MCH PUs and LUs in the X.25 NPSI server domain.
- Define the switched circuits in the X.25 NPSI domain.

X.25 NPSI Server Configuration Statements

Following are the syntax and description of the valid statements used in the data set pointed to by the //X25IPI DD statement in your X.25 NPSI cataloged procedure.

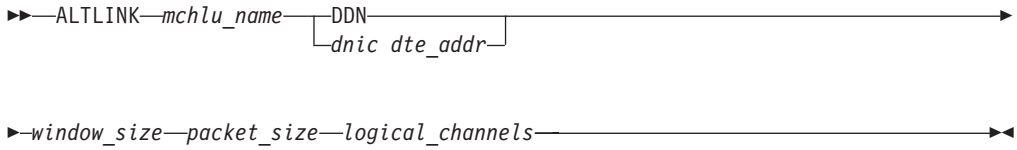
Statement Syntax

- Each statement is on a separate line in the configuration data set.
- Each statement starts with the keyword followed by the parameter fields, separated by one or more blanks.
- The statements are not case-sensitive. You can enter them in both upper and lower case.
- Comment lines are marked with an asterisk in column 1.

ALTLINK Statement

Use the ALTLINK statement to specify members of a hunt group for incoming calls. The collection of lines in this group are assigned a single X.25 address. Incoming X.25 calls will be accepted from any of the lines in the group and outgoing calls will be rotated across the lines. If one of the lines is not operational, outgoing calls will be rotated on to the next available line in the group.

Syntax



Parameters

mchlu_name
The name of the physical circuit logical unit (NPSI MCH LU).

DDN
Use DDN for the Defense Data Network.

dnic
The X.121 Data Network Identifier Code (DNIC) for the public data network. *dnic* can be coded as PRIVATE or PRIV to denote a private X.25 network.

dte_addr
The X.25 DTE address for the link. The address must be between 1 and 15 decimal digits. This parameter is not coded for DDN links. Specify *dte_addr* as NONE to omit the calling address from the call request packet.

window_size
The window size to negotiate on switched virtual circuits, in the range of 1 to 7 for a modulo-8 network, or 1 to 127 for a modulo-128 network.

packet_size
Choose one of the following X.25 packet sizes as the default: 32, 64, 128, 256, 512, 1024, 2048, or 4096 bytes.

logical_channels
The number of logical channels (switched virtual circuits) subscribed, in the range of 1 to 1023.

Examples

The following example shows the LINK, ALTLINK, FAST, and OPTIONS statements for a public network connection with two-line hunt group:

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU021     3110 23456789        2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890        2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
```

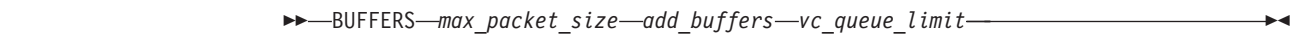
Usage Notes

- The ALTLINK statement must follow a LINK statement.
- A DEST statement follows the last ALTLINK statement for a hunt group.
- If special options are required or fast connect is used, OPTIONS or FAST statements must immediately follow the ALTLINK to which they apply.

BUFFERS Statement

Use the BUFFERS statement to specify the buffer size for IP datagrams.

Syntax



Parameters

max_packet_size
The maximum IP packet size. This value must match the *max_packet_size* parameter on the GATEWAY statement in the *hlq*.PROFILE.TCPIP data set, and must be in the range of 576 to 2048.

add_buffers
The number of additional buffers to allocate, in addition to the minimum of 2 for each logical channel.

vc_queue_limit
The limit on the number of buffers queued outbound on any single virtual circuit.

Examples

The following statement specifies a maximum IP packet size of 1024, an allocation of 50 additional buffers, and a limit of 4 queued outbound buffers on any SVC:

```
*
Buffers  1024      50        4
*
```

Usage Notes

- This maximum IP packet size must be at least as large as the largest *max_packet_size* parameter on the GATEWAY entries for X.25 NPSI LINKS in the *hlq*.PROFILE.TCPIP data set.
- Additional buffers are required for coping with traffic peaks and holding outbound IP datagrams while new X.25 connections are being established. Use a larger value when many X.25 destinations are called in a short period of time.

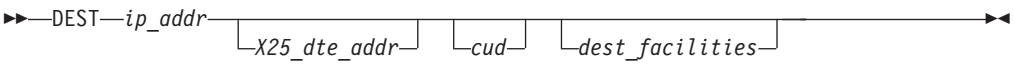
Related Topics

- “GATEWAY Statement” on page 193
- “OPTIONS Statement” on page 377
- *OS/390 SecureWay Communications Server: IP Diagnosis*

DEST Statement

Use the DEST statement to specify the destination address list for the link.

Syntax



Parameters

ip_addr
The IP address in dotted decimal format. At least 1 byte must be supplied; omitted trailing bytes are not checked when determining a match.

X25_dte_addr
The corresponding X.25 DTE destination address for the link (1 to 15 decimal digits). Do not code this parameter for Defense Data Network (DDN) destinations.

cud
The call user data (CUD) protocol identifier used. A hexadecimal number with a default value of X'CC'. Do not code this parameter for DDN destinations.

dest_facilities
The X.25 facilities field to be used on outgoing calls for this destination. This value overrides the FACILITIES value in the OPTIONS statement for this destination. Specify this value as an even number of hexadecimal digits. The field is inserted in outgoing call packets following facilities generated from window or packet size negotiation or reverse charging. The facilities length byte is calculated automatically and should not be coded here.

Examples

The following example shows the LINK and DEST statement for a DDN connections.

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- ---------------- -      ----   ---
Link     XU023     DDN                  2       128   16
*
*        IP address      X.25 DTE addr
*        --------------- ---------------
Dest     10
*
```

Usage Notes

- The DEST statement must follow the LINK statement.
- Data Defense Network destinations do not use the X.25 DTE address and the CUD protocol identifier.

FAST Statement

Use the FAST statement to provide NPSI fast connect for links with heavy activity. Fast connect is only used for SVCs connected to non-SNA data terminal equipment (DTE). See *X.25 NPSI Host Programming* for more information.

Parameters

prefix
The fast connect VC LU name prefix. This is the MCH LU name unless the PRFLU option is coded on the NPSI X25.VC statement.

HEX or DEC
The fast connect VC LU numbering scheme (if the HEXNAME parameter is coded in the NPSI X25.VC statement). The default is HEX.

suffix
The fast connect VC LU numbering base (if the SUFFIX parameter is coded in the NPSI X25.VC statement). The default is 001.

Examples

The following example shows the placement of a FAST statement that specifies a *prefix* of XU021 and takes the default values of HEX and 001.

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- ---------------- -      ----   ---
LINK     XU021     3110 23456789        2      128    16
FAST     XU021
OPTIONS  GATE
*
```

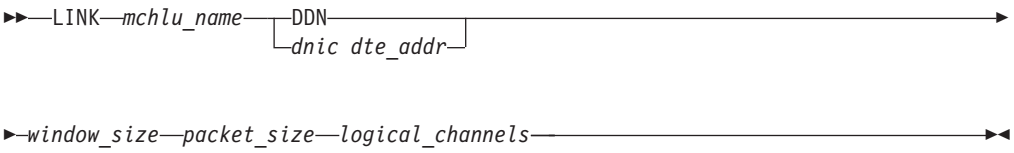
Usage Notes

- The OPTIONS GATE statement is required.
- The FAST statement must follow the LINK or ALTLINK statement to which it applies.
- The *prefix* value must match the value in your NPSI configuration.

LINK Statement

Use the LINK statement to define the NPSI MCH LU names. One SNA control session is established for each MCH LU defined by a LINK statement.

Syntax



Parameters

mchlu_name
The name of the physical circuit logical unit (NPSI MCH LU).

DDN
Use DDN for the Defense Data Network.

dnic
The X.121 Data Network Identifier Code (DNIC) for the public data network. *dnic* can be coded as PRIVATE or PRIV to denote a private X.25 network.

dte_addr
The X.25 DTE address for the link. The address must be between 1 and 15 decimal digits. This parameter is not coded for DDN links. Specify *dte_addr* as NONE to omit the calling address from the call request packet.

window_size
The window size to negotiate on switched virtual circuits, in the range of 1 to 7 for a modulo-8 network, or 1 to 127 for a modulo-128 network.

packet_size
Choose one of the following X.25 packet sizes as the default: 32, 64, 128, 256, 512, 1024, 2048, or 4096 bytes.

logical_channels
The number of logical channels (switched virtual circuits) subscribed, in the range of 1 to 1023.

Examples

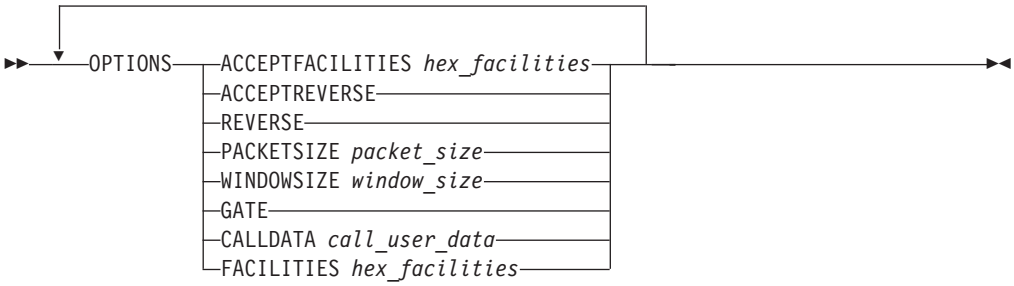
In the following example, AU20 is the name of a non-DDN network, and AU16 is the name of a DDN network.

```
*        NPSI MCH        DTE             Window Packet Logical
*        LU Name   DNIC  Address         Size   Size   Channels
*        --------  ----  --------------  -      ----   ---
LINK     AU20      3020  90201234548      2     1024    5
LINK     AU16      DDN                    2     1024    5
*
```

OPTIONS Statement

Use the OPTIONS statement to specify the call handling options for each link. Values specified on the OPTIONS statement apply to all outgoing calls on the LINK MCH, but can be overridden for individual destination addresses by the DEST statement. More than one OPTIONS statement can be coded after each LINK statement, Several parameters can be placed in a single OPTIONS statement but cannot continue on the next line. If all the parameters do not fit on one line, use additional OPTIONS statements.

Syntax



Parameters

ACCEPTFACILITIES *hex_facilities*
The X.25 facilities field to be used when accepting incoming calls. Specify this value as an even number of hexadecimal digits. The facilities length byte is calculated automatically and should not be coded here.

ACCEPTREVERSE
Causes incoming calls with the reverse charging facility to be accepted. The default action is to clear reverse charge calls.

REVERSE
Includes the reverse charging facility in all outgoing call request packets.

PACKETSIZE *packet_size*
The packet size to negotiate on switched virtual circuits, one of the values 32, 64, 128, 256, 512, 1024, 2048, or 4096 bytes.

WINDOWSIZE *window_size*
The window size to negotiate on switched virtual circuits, in the range of 1 to 7 for a modulo-8 network, or 1 to 127 for a modulo-128 network.

GATE
Specified if the NPSI MCH is defined with GATE=GENERAL to permit sharing of an X.25 physical link with other services.

CALLDATA *call_user_data*
The call user data field to be used on outgoing X.25 calls, an even number of hexadecimal digits. The standard value for IP traffic must begin with the protocol identifier CC.

FACILITIES *hex_facilities*
The X.25 facilities field to be used on outgoing calls for this destination. Specify this value as an even number of hexadecimal digits. The field is inserted in outgoing call packets following facilities generated from window or packet size

negotiation or reverse charging. The facilities length byte is calculated automatically and should not be coded here.

Examples

The following example shows the proper placement of the OPTIONS statements when using both LINK and ALTLINK statements.

```
Link     XU021     3110 23456789        2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890        2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
*
```

Usage Notes

- The OPTIONS statements must follow the LINK or ALTLINK statements to which they apply.
- Negotiation takes place on outgoing calls if the window size or packet size on the OPTIONS statement is different from the network defaults coded in the LINK statement.
- The *max_packet_size*, also called the maximum transmission unit (MTU), coded in the BUFFERS statement must be large enough to hold the largest IP datagram to be transmitted or received over the link. If the MTU is greater than the X.25 packet size, an IP datagram is sent as an X.25 packet sequence. The buffer size must be sufficient to hold the combined data of the sequence. The MTU for DDN networks is 1007. See RFC 877 for more information. Information on how to obtain RFCs is included in “Appendix D. Related Protocol Specifications (RFCs)” on page 1177

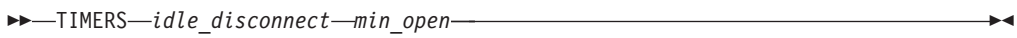
Related Topics

- “GATEWAY Statement” on page 193
- “BUFFERS Statement” on page 373

TIMERS Statement

Use the TIMERS statement to specify the limits for various timers.

Syntax



Parameters

idle_disconnect
Time, in seconds, after which an idle or inactive connection is cleared.

min_open
The minimum time, in seconds, a connection is held before it can be preempted for a new destination.

Examples

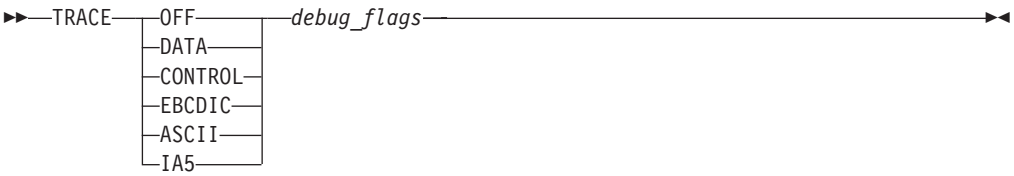
The following statement clears inactive connections after 5 minutes and holds a connection open for 1 minute.

```
*        Idle      Minimum
*        Disconn.  Open
*        --------  -------
Timers   300       60
*
```

TRACE Statement

Syntax

Use the TRACE statement to specify the trace and debug levels for the X.25 NPSI server. The trace and debug functions are independent of one another. You can turn tracing off and still request debug options.



Parameters

OFF
Turns tracing off. If you specify OFF, data on the connection is not traced.

DATA
Traces the data packets on the connection and displays the full contents of the IP datagrams. This is equivalent to the ASCII option.

CONTROL
Traces the data packets on the connection and displays only the X.25 control packet.

EBCDIC
Traces the data packets on the connection and displays the data in EBCDIC.

ASCII
Traces the data packets on the connection and displays the data in ASCII.

IA5
Traces the data packets on the connection and displays the data in IA5.

debug_flags
A string of eight positional flags that control the display of debugging information. Each flag has the value of 1 or 0, where 1 turns the flag on and 0 turns the flag off. The flags are:

Position	Description
0	Display configuration records
1	Display commands
2	Trace DLC events
3	Trace VTAM events
4	Display control block addresses
5	Main loop dispatching
6	Reserved for internal use
7	Send information and warning messages to the operator's console

Examples

The following statement turns tracing off and sets two debug flags:
Value 1 in position 0
Displays configuration records

Value 1 in position 7
Sends information and warning messages to the operator's console

```
*                 01234567
Trace    OFF      10000001
*
```

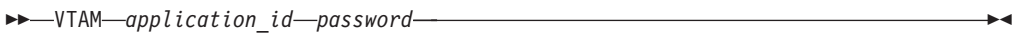
Related Topics

OS/390 SecureWay Communications Server: IP Diagnosis

VTAM Statement

The VTAM statement allows you to access the VTAM definition for the application. The VTAM statement must precede the LINK statement.

Syntax



Parameters

application_id
The application identifier in the VTAM APPL definition. This is either the name specified in the first 8 columns or the ACBNAME if one is defined.

password
The password for the VTAM application specified in the VTAM APPL definition.

Examples

This VTAM statement is correct for either of the VTAM APPL definitions that follow it.

```
*--
VTAM     TCPIPX25  TCPX25
*--
```

VTAM APPL definitions:

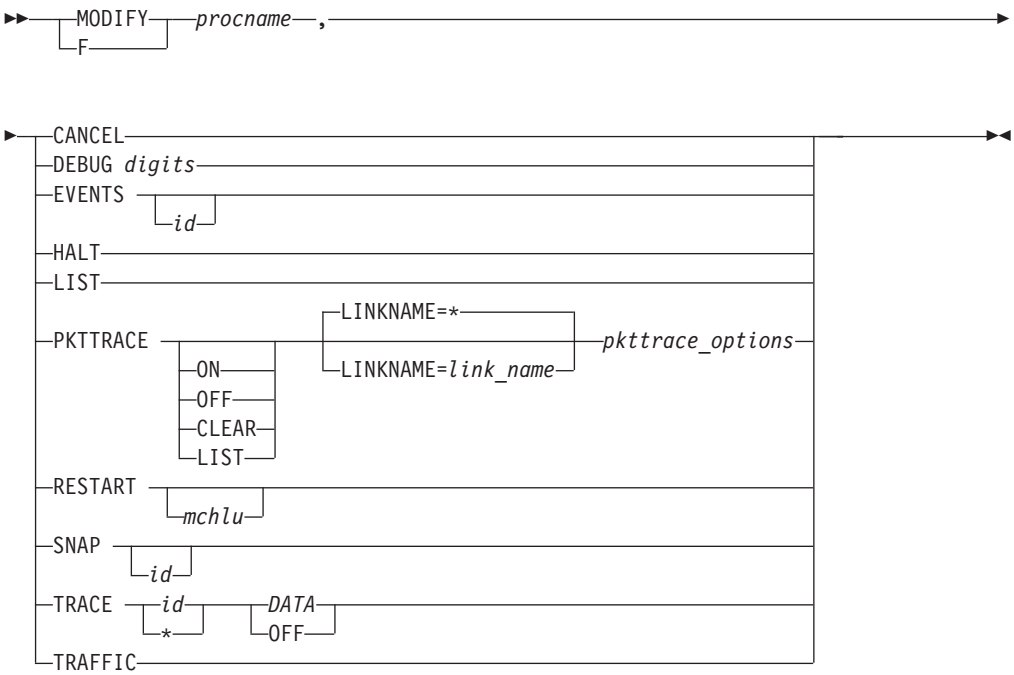
```
X25APPL2 APPL  ACBNAME=TCPIPX25,
               PRTCT=TCPX25,
               AUTH=(ACQ),
               PARSESS=YES,
               EAS=20

TCPIPX25 APPL  PRTCT=TCPX25,
               AUTH=(ACQ),
               PARSESS=YES,
               EAS=20
```

Operating the X.25 NPSI Server Using the MODIFY Command: You can operate the X.25 NPSI server from the operator's console using MODIFY command. Following is the correct syntax and valid parameters for the X.25 NPSI server.

MODIFY Command—X.25 NPSI Server

Use the MODIFY command to pass parameters to the X.25 NPSI server.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start this server.

CANCEL
Cancels the X.25 NPSI server task and produces a dump.

DEBUG *digits*
Alters debug settings, where *digits* is a string of debug levels corresponding to those in the configuration data set for X.25 NPSI server.

EVENTS *id*
Displays event handler names for debugging, where *id* is an optional LU name or logon ID.

HALT
Shuts down the X.25 NPSI task, closing all connections.

LIST
Displays a list of the status of the virtual circuit.

PKTTRACE
Enables or disables tracing of IP packets and modifies the selection criteria for screening packets in the X.25 NPSI interface. The *pkttrace_options* options are:

- ABBREV
- DESTPORT
- FULL
- IP

- LINKNAME
- PROT
- SRCPORT
- SUBNET

For explanations of the PKTTRACE options, see “PKTTRACE Statement” on page 223.

RESTART *mchlu*
Attempts to reacquire failed links (MCHs), after reactivating them through VTAM. Where *mchlu* is an optional LU name from a link definition. If omitted, all inactive MCHs are restarted.

SNAP *id*
Displays program data areas for debugging, where *id* is an optional LU name or logon ID.

TRACE
Alters the trace level, where *id* is an optional LU name, logon ID, or an asterisk (*). TRACE can be one of two levels: DATA or OFF.

TRAFFIC
Displays traffic counts.

Examples

To halt an X.25 NPSI server whose procedure started with the following statements in the *hlq*.PROFILE.TCPIP

```
AUTOLOG
   TCPIPX25
```

you could issue either of these commands at the operator’s console:

```
MODIFY TCPIPX25,HALT
```

```
F TCPIPX25,HALT
```

Usage Notes

TCP/IP for MVS allows the configuration of multiple DLC links to the SNALINK LU0, LU6.2, and X.25 NPSI server address spaces. The PKTTRACE parameter supports this capability through the LINKNAME parameter. Therefore, multiple PKTTRACE parameters can be issued to define the scope of the tracing by identifying the tracing options applicable to multiple links.

PKTTRACE considerations:

- Parsing of the parameter halts as soon as an error is detected and the parameter is ignored.
- Parameters can appear in any order.
- The occurrence of a parameter more than once is an error. In the case of the special parameters ON, OFF, CLEAR, and LIST, the occurrence of more than one of these parameters is an error.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created, or reset using the CLEAR parameter, a link’s trace profile is set to the default values for the trace parameters as follows:

  DESTPORT
  No checking

FULL Tracing of the whole IP packet

IP All IP addresses (*)

PROT All protocols (*)

SRCPORT
No checking

SUBNET
No checking

- Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (*), which indicates all links. When multiple statements refer to the same link, the parameters on the statements are cumulative, and parameters not specified on the second and subsequent statements are not changed. If a parameter is specified on one statement and then appears on a subsequent statement, the value associated with the last occurrence of the option is used because this is the value that is stored in the trace profile.

Chapter 11. Configuring the OS/390 UNIX Telnet Server

This chapter contains information about installing, starting, stopping, and administering OS/390 UNIX Telnet.

The OS/390 UNIX Telnet server described in this chapter provides access to OS/390 UNIX shell applications on the host using the Telnet protocol. The Telnet server described in “Chapter 12. Configuring the Telnet Server” on page 395 provides access to SNA applications on the host using Telnet TN3270E, TN3270, or linemode protocol.

Installation Information

The HFS files used in the OS/390 UNIX Telnet server and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file. otelnetd writes to syslog facility local1.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/otelnetd
This is a symbolic link to /usr/lpp/tcpip/sbin/otelnetd. /usr/lpp/tcpip/sbin/otennetd is a sticky-bit file. The member of OTENNETD of hlq.SEZALINK contains the executable code for the Telnet server.

/etc/banner
This file contains a login message which will be printed to the client’s screen unless the -h option is specified. This banner should be stored here.

/bin/fomtlinp
The executable file for utmp entry is stored here. This code will update the utmp entry as well as generate the child.

/etc/utmpx
This file is updated by the call to fsumoclp. It contains a list of all the users who are logged in with their associated tty.

/dev/ptypXXXX and /dev/ttypXXXX
These special device files represent pseudoterminals (ptys); they are used by the OS/390 UNIX Telnet server and other programs.

Note: For information on allocating more of these files for more connections, see *OS/390 UNIX System Services Planning*.

/usr/share/lib/terminfo
The descriptions of supported terminals are stored here. For more information, see *OS/390 UNIX System Services Planning*.

/usr/lib/nls/msg/C/tnmsgs.cat
The message catalog used by the OS/390 UNIX Telnet server is stored here.

Where the server looks for the message catalog depends on the value of NLSPATH and LANG environment variables. If you want to store the message catalog elsewhere, you need to change the NLSPATH or the LANG environment variables. If the message catalog does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

Note: The OS/390 UNIX inetd daemon does not propagate environment variables other than PATH and TZ to its child processes.

/usr/man/C/cat1/otelnetd.1
This file contains the associated manual (man) pages for the OS/390 UNIX Telnet server. It provides online help for the user.

Starting, Stopping, and Administration of OS/390 UNIX Telnet

The OS/390 UNIX Telnet server is started by inetd for each incoming Telnet connection. When the Telnet session is complete, the OS/390 UNIX Telnet server will exit. Each active Telnet session will have a separate instance of the Telnet server which will communicate with the Telnet client.

The following standards are supported:
- RFC 854 Telnet Protocol Specification
- RFC 855 Telnet Option Specification
- RFC 856 Telnet Binary Transmission
- RFC 857 Telnet Echo Option
- RFC 858 Telnet Suppress Go Ahead Option
- RFC 859 Telnet Status Option
- RFC 860 Telnet Timing Mark Option
- RFC 861 Telnet Extended Options - List Option
- RFC 885 Telnet End of Record Option
- RFC 1073 Telnet Window Size Option
- RFC 1079 Telnet Terminal Speed Option
- RFC 1091 Telnet Terminal type option
- RFC 1096 Telnet X Display Location Option
- RFC 1123 Requirements for Internet Hosts -- Application and Support
- RFC 1184 Telnet Linemode Option
- RFC 1372 Telnet Remote Flow Control Option
- RFC 1571 Telnet Environment Option Interoperability Issues
- RFC 1572 Telnet Environment Option

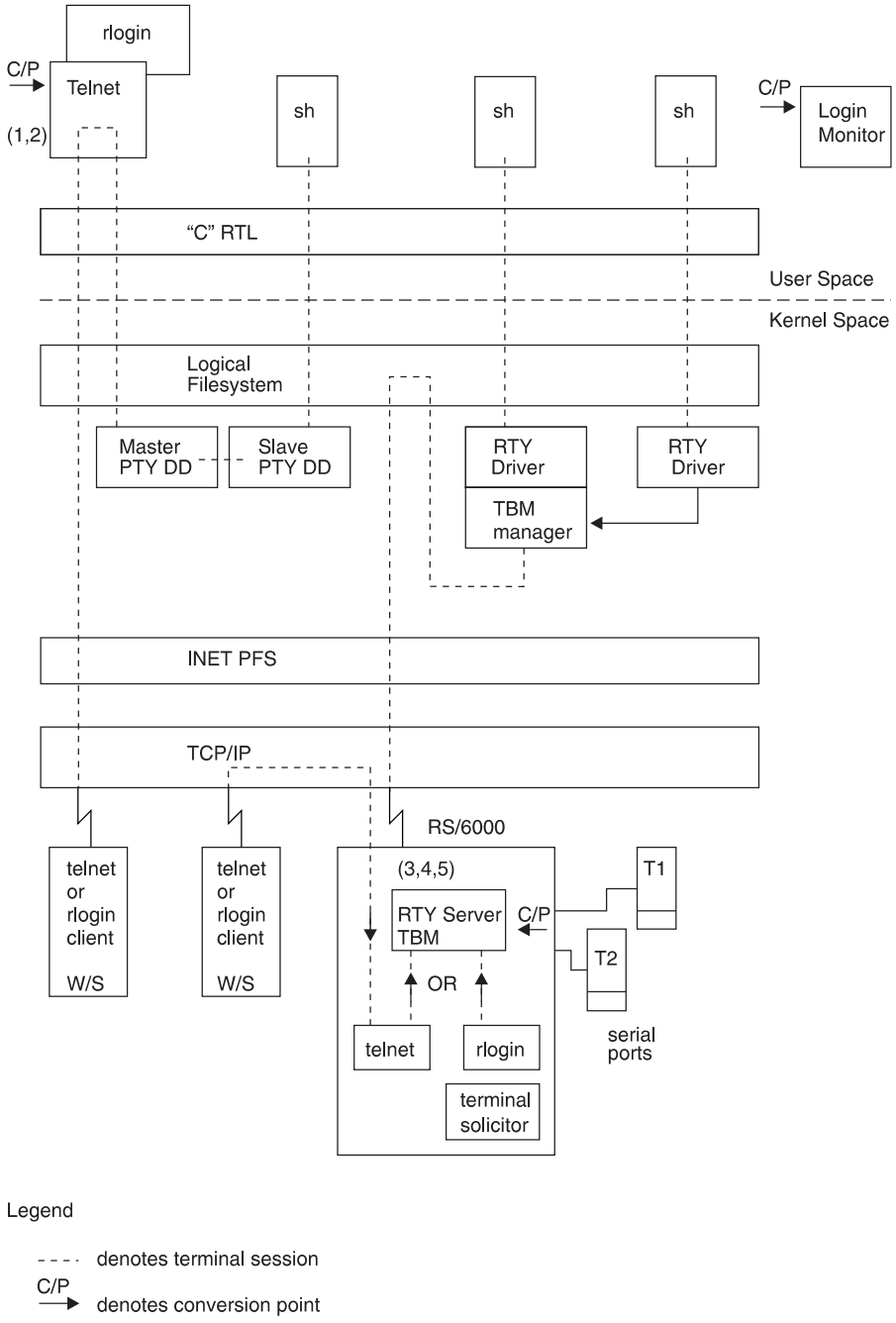


Figure 18. OS/390 UNIX Terminal Attachment Paths. The following terminal attachment paths are illustrated:

- *Paths 1 and 2 are Telnet and rlogin paths from remote clients to servers located on the Open Edition host system.*
- *Paths 3, 4 and 5 are attachment paths provided by OCS and Telnet and rlogin servers are outbound on the RISC System/6000. Serial terminal attachment is also provided by the RISC System/6000.*

When an OS/390 UNIX Telnet session is started up, otelnetd sends Telnet options to the client side indicating a willingness to do the following:

- DO TERMINAL TYPE
- DO TSPEED

- DO XDISPLOC
- DO NEW-ENVIRON
- DO ENVIRON
- WILL SUPPRESS GO AHEAD
- DO ECHO
- DO LINEMODE
- DO NAWS
- WILL STATUS
- DO LFLOW
- DO TIMING-MARK

With the following OS/390 UNIX Telnet options, Telnet has support for enabling **LOCALLY.**

- WILL BINARY

  This option indicates that the client is willing to send 8 bits of data, rather than the normal 7 bits of network virtual terminal data.
- WILL ECHO

  When the LINEMODE option is enabled, a WILL ECHO or WONT ECHO will be sent to the client to indicate the current state of terminal echoing. When terminal echo is not desired, a WILL ECHO is sent to indicate that Telnet will take care of echoing any data that needs to be echoed to the terminal, and then nothing is echoed. When terminal echo is desired, a WONT ECHO is sent to indicate that Telnet will not be doing any terminal echoing, so the client should do any terminal echoing that is needed.
- WILL LOGOUT

  When a DO LOGOUT is received, a WILL LOGOUT is sent in response and the Telnet session is shut down.
- WILL SGA

  This option indicates that it will not be sending IAC GA, the go ahead command.
- WILL STATUS

  Indicates a willingness to send the client, upon request, the current status of all Telnet options.
- WILL TIMING-MARK

  Whenever a DO TIMING-MARK is received, a WILL TIMING-MARK is the response. It is only used in kludge linemode support.

With the following OS/390 UNIX Telnet options, Telnet has support for enabling **REMOTELY.**

- DO BINARY

  Sent to indicate that Telnet is willing to receive an 8-bit data stream.
- DO ECHO

  If a WILL ECHO is received, a DONT ECHO will be sent in response.
- DO ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1408.)
- DO LFLOW

  Requests that the client handle flow control characters remotely.
- DO LINEMODE

  Supports requests that the client do line-by-line processing.
- DO NAWS

Requests that the client inform the server when the window size changes.

- DO NEW-ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1572.)

- DO SGA

  Indicates that it does not need to receive IAC GA, the go ahead command.

- DO TERMINAL-TYPE

  Indicates a desire to be able to request the name of the type of terminal that is attached to the client side of the connection.

- DO TERMINAL-SPEED

  Indicates a desire to be able to request information about the speed of the serial line to which the client is attached.

- DO TIMING-MARK

  Only supported if the client responded with WONT LINEMODE. If the client responds with WILL TM, then it is assumed that the client will support kludge linemode. It is not used for any other purposes.

- DO XDISPLOC

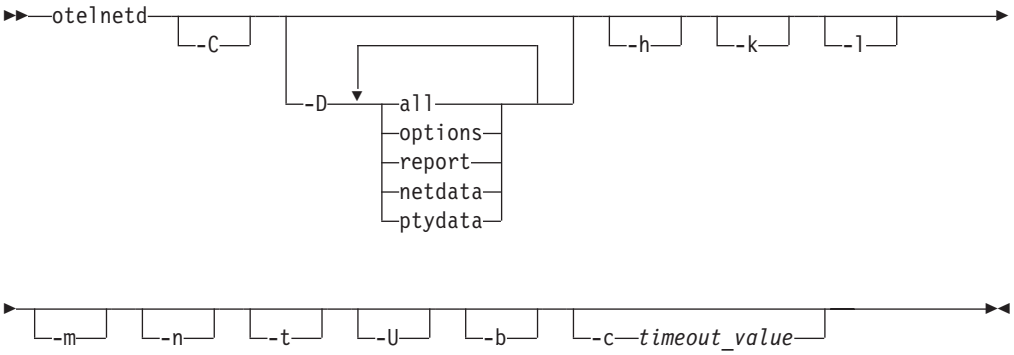
  Indicates a desire to be able to request the name of the X Window System display that is associated with the Telnet client.

otelnetd

Note: The userid associated with the daemon in /etc/inetd.conf requires superuser authority. See ″Setting Up for Daemons″ in *OS/390 UNIX System Services Planning* for a description of the kinds of authority defined for daemons.

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke otelnetd.

Syntax



Parameters

-C Prints user messages in uppercase. There are several exceptions. Messages issued on startup are not affected by the -C option because the -C option is not processed during the startup. Also, data transmittal messages will not be uppercase. Data transmittal messages are the output from the -D netdata option or the -D ptydata option.

-D The -D option has several suboptions:

options
Prints information about the negotiation of Telnet options. This is used for debugging purposes. It allows telnetd to print out debugging information to the connection, allowing the user to see what telnetd is doing.

report Prints the options information, plus some additional information about what processing is going on. This also includes print information slated for suboption=options. This is used for debugging purposes. It allows telnetd to print out debugging information to the connection, to enable the user to see what telnetd is doing.

netdata
Displays the data stream received by telnetd. This is used for debugging purposes. It allows telnetd to print out debugging information to the connection, to enable the user to see what telnetd is doing.

ptydata
Displays the data stream written to the pty. This is used for debugging purposes. It allows telnetd to print out debugging information to the connection, to enable the user to see what telnetd is doing.

all Enables options, report, netdata, and ptydata.

-h Disables the display of the /etc/banner file at the user's terminal.

-k

Disables kludge linemode. The server normally attempts to use kludge linemode when the -l option was specified, but the client does not support line mode. Use the -k option when there are remote clients that do not support kludge linemode, but pass the hueristic for kludge line mode support (for example, if they respond with WILL TIMING-MARK in response to a DO TIMING-MARK). This option does not disable kludge line mode when the client requests it. This is done by the client sending DONT SUPPRESS-GO-AHEAD and DONT ECHO.

-l Specifies line mode; tries to force clients to use line mode. If the LINEMODE option is not supported and the -k option was not specified, it will try to use kludge linemode.

Notes:

1. Many clients decline the server's request to operate in linemode.
2. Linemode is not appropriate for full-screen applications like the OS/390 UNIX vi editor.

-m Enables creation of a forked or spawned process to coexist in the same address space. Using this option improves performance.

-n Disable TCP keep-alives. Normally telnetd enables the TCP keep-alive mechanism to probe connections that have been idle for some period of time to determine if the client is still there, so that idle connections from machines that have crashed or can no longer be reached can be cleaned up. The cleanup of disabled connections is controlled by the presence of the KEEPALIVEOPTIONS statement in the TCPIP profile.

-t Internal tracing. It will also turn on the REPORT option, as if the user also specified -Dd Report.

-U This option causes telnetd to drop connections from any IP address that cannot be mapped back into a symbolic name via the gethostbyaddr(3) routine.

-b This option forces the server to DO BINARY in the first pass during negotiations with the client.

-c *timeout_value*
Specifies the number of seconds to wait before terminating the telnet session if there is no activity on the connection. The *timeout_value* can be between 1 and 86400 seconds.

SMF Record Handling

The SMF records generated are the typical set of records that MVS generates for start of job (login) and end of job (logoff). Additionally, interval records can be issued during the life of the user login. These records are SMF TYPE 30 and TYPE 72 and not the TYPE 118 in the current Telnet server. The process of issuing these records is external to the specific daemons.

BPX.DAEMON Considerations

If the BPX.DAEMON facility class is defined, the following additional configuration must be performed:

1. The userid specified in /etc/inetd.conf for otelnetd must have read access to BPX.DAEMON.
2. hlq.SEZALINK must be defined to program control.
3. The C run-time library, hlq.SCEERUN, must be defined to program control.

See UNIX System Services Planning for more information about the BPX.DAEMON facility class, the security product commands used to perform the required configuration, and the diagnosis procedure for resolving related problems.

Chapter 12. Configuring the Telnet Server

Telnet is a terminal emulation protocol that allows you to log on to a remote host as though you were directly attached to that host. Telnet allows users on any host to have access to applications running on that host. With Telnet, you can establish concurrent sessions on different hosts or multiple sessions with a single host.

The Telnet server runs in the TCPIP address space. It is a part of TCP/IP. It does not have its own start-up procedure: it is started when TCP/IP is started. The Telnet server communicates with VTAM using either LU0 or LU2 for terminal support and LU1 and LU3 for printer support.

When TCP/IP is started, the program loads Telnet server configuration information from a PROFILE data set. Once TCP/IP is started, you can use a special set of VARY and DISPLAY commands specifically related to Telnet server functions to alter the state of Telnet or display information about Telnet connections. For information about these commands sets, see “Chapter 13. Managing the Telnet Server” on page 489.

This chapter describes the configuration options that you have when you configure the Telnet server. For example, this release supports up to 255 Telnet ports. This chapter also provides the information required to make configuration decisions about using the Telnet server in your environment.

The Telnet server described in this chapter provides access to SNA applications on the host using Telnet TN3270E, TN3270, or linemode protocol. The OS/390 UNIX Telnet server described in “Chapter 11. Configuring the OS/390 UNIX Telnet Server” on page 387 provides access to OS/390 UNIX shell applications on the host using the Telnet protocol.

Configuration Options

When you configure the Telnet server, you can specify these options.

- “Customizing the TCPIP Cataloged Procedure” on page 396

  If you want to provide full-screen access to Telnet from non-3270 terminals to support the VT100 and VT282 families of terminals, then you need to configure the Telnet server for 3270 single byte character set (SBCS) or double byte character set (DBCS) transform mode. This requires that you include special data definition (DD) statements in the TCPIP cataloged procedure and specify the TRANSFORM or DBCSTRANSFORM parameter on the TELNETPARMS statement in the TCPIP PROFILE data set.
- “Customizing the VTAM Configuration Data Set” on page 397

  To ensure that the VTAM applications and logical units (LUs) needed by Telnet are started when VTAM is started, you need to update the VTAM configuration data set.
- “Customizing Statements in the PROFILE Data Set” on page 398

  During initialization of the TCP/IP address space, system operation and configuration parameters are read from a configuration PROFILE data set. There are several kinds of information you specify in the PROFILE data set:

  – “Specifying the Stand-Alone PORT Statement” on page 402

There are two ways in the PROFILE data set to specify which port will be used by Telnet: using the PORT statement under TELNETPARMS and using the stand-alone PORT statement with the INTCLIEN keyword that reserves a port for Telnet. In Version 3.2 of TCP/IP for MVS, the stand-alone PORT statement was required. The stand-alone PORT statement is now optional, but recommended.

  - "Specifying the TELNETPARMS Statements in the PROFILE Data set" on page 404

    If you want to specify parameters for the Telnet server, update the TELNETPARMS information block within the PROFILE data set. This information block was called INTERNALCLIENTPARMS in earlier releases. You can specify such Telnet server characteristics as the port that Telnet will use, the time intervals between scanning for idle connections, or the codepage used to support linemode connections.

  - "Specifying BEGINVTAM Statements in the PROFILE Data Set" on page 429

    If you want to specify VTAM parameters for configuring the Telnet server, use the BEGINVTAM information block within the PROFILE data set. You can find the general rules for VTAM statements in "General Rules for PROFILE Statements" on page 400.

- "Telnet PROFILE Example" on page 457

  If you want to see how these statements work in an actual PROFILE, look at this example.

Customizing the TCPIP Cataloged Procedure

If you want to configure the Telnet server for 3270 SBCS transform, use the TRANSFORM parameter to load the 3270 transform module, TNSIMHPI, at initialization. This module is only available as a third-party product.

If you want to configure the Telnet server for 3270 DBCS transform mode, use the DBCSTRANSFORM parameter and update the TCPIP cataloged procedure. You can find a sample of this procedure and instructions for setting it up in "Update the TCP/IP Cataloged Procedure" on page 97.

To configure the Telnet server for 3270 DBCS transform mode, include the following DD statements in the TCPIP cataloged procedure:

```
//TNDBCSCN DD DSN=TCPIP.V2R7.SEZAINST(TNDBCSCN),DISP=SHR
//TNDBCSXL DD DSN=TCPIP.V2R7.SEZAXLD2,DISP=SHR
//TNDBCSER DD SYSOUT=A
```

In this example:

- The TNDBCSCN DD statement must point to the configuration data set for 3270 DBCS transform mode. This configuration data set specifies the default DBCS conversion mode that will take effect at initialization time.

  Specify the CODEKIND and CHARMODE parameters according to the required DBCS code page. If CODEKIND and CHARMODE are not specified, or if the TNDBCSCN DD statement is not configured, CODEKIND defaults to SJISKANJI and CHARMODE defaults to ALPHABET.

  The following sample configuration can be found in the installation data set TCPIP.V2R7.SEZAINST(TNDBCSCN).

```
#
#*********************************************************************
#*                                                                   *
#*  TNDBCSCN - DBCS Transform Mode TELNET Configuration Data Set     *
#*                                                                   *
#*  Usage of this data set.                                          *
#*                                                                   *
#*   CODEKIND : default DBCS code page.                              *
#*              1. SJISKANJI  --- Shift JIS Kanji                    *
#*              2. JIS83KJ    --- '83 JIS Kanji                      *
#*              3. JIS78KJ    --- '78 JIS Kanji                      *
#*              4. DECKANJI   --- DEC Kanji                          *
#*              5. EUCKANJI   --- EUC Kanji                          *
#*              6. KSC5601    --- Korean KSC5601 Hangeul             *
#*              7. HANGEUL    --- Korean Original Hangeul            *
#*              8. BIG5       --- BIG5                               *
#*              9. SCHINESE   --- Simplified  Chinese                *
#*             10. TCHINESE   --- Traditional Chinese - 5550         *
#*                                                                   *
#*   CHARMODE : default basic character mode.                        *
#*              (For use with Japanese code pages only.)             *
#*              1. ALPHABET --- alphabet mode                        *
#*              2. KATAKANA --- katakana mode                        *
#*                                                                   *
#*                                                                   *
#*********************************************************************
#
CODEKIND = SJISKANJI
CHARMODE = ALPHABET
```

- The TNDBCSXL DD statement must point to the data set containing binary translation table codefiles for 3270 DBCS transform mode. The installation data set, *hlq*.SEZAXLD2, contains the default binary translation table codefiles.
- DBCSTRACE sends debug output from 3270 DBCS transform mode to the location specified in the SYSPRINT output DD statement.

Customizing Translate Tables for 3270 DBCS Transform Mode

You can customize the binary translation table codefiles for 3270 DBCS transform mode using the CONVXLAT command. See “Converting Translation Tables to Binary” on page 1144 for more information on customizing DBCS translation table codefiles.

Customizing the VTAM Configuration Data Set

To ensure that the VTAM Telnet applications are started when VTAM is started, you need to update the VTAM configuration data set.

You can do this by copying the sample provided in *hlq*.SEZAINST(VTAMLST) to the ATCONNxx member of your VTAMLST and modifying it to suit your installation. This member contains the VTAM Telnet applications and ensures that these applications will be started when VTAM is started.

When you update the VTAM configuration data set, you can define Telnet LUs to VTAM using a wildcard character, rather than coding individual Telnet application LU statements. VTAM Version 4 Release 3 introduced a new function called “model application names.” This function gives system administrators the ability to code a generic APPL name using the “*” character. This is significant for Telnet

administrators because in the past, each Telnet LU that represented a Telnet connection required its own individual application definition statement.

If, for example, you defined 20,000 Telnet users, you could use a range statement to code these on the TCP/IP side (that is, LU00001..LU20000), but on the VTAM side, you had to code individual APPL statements for each of these 20,000 LUs. The ″model application″ VTAM function lets you code these using only one statement (that is, LU* APPL....).

With VTAM release 4.3 or higher, an asterisk (*) can be used as a wild card character to replace a character string at the same position anywhere within the minor node name. In the following sample VTAMLST, `TCP*` is used to represent all TCP entries

Below is a sample VTAMLST. This sample does not take into account the continuation character in column 72.

```
*  VTAMLST Sample Definition
*
*  COPYRIGHT = NONE.
*
TCP      VBUILD TYPE=APPL
TCP*     APPL   AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,SESSLIM=YES
```

Because the TCP/IP LU code cannot handle multiple concurrent sessions, you must code SESSLIM=YES for the TELNET LUs defined to VTAM. Otherwise, if SESSLIM=NO, menu or session manager applications that use “return session” processing might cause session termination.

Customizing Statements in the PROFILE Data Set

During initialization of the TCPIP address space, system operation and configuration parameters are read from a configuration PROFILE data set. The PROFILE data set is used to configure Telnet to accept or reject connection requests. You can update the PROFILE data set to change or add statements to support new functions, or change or add usage rules. You could modify the PROFILE data set, for example, to define which VTAM LUs are associated with which IP addresses or host names, which host printers are defined for the Telnet client, or which ports are used by Telnet.

There are three sets of parameters in the PROFILE data set that you can edit and customize. These are:

1. The stand-alone PORT statement with the INTCLIEN keyword (optional)
2. TELNETPARMS (INTERNALCLIENTPARMS) information block for defining PROFILE statements relating to Telnet port setup
3. BEGINVTAM information block for defining PROFILE statements relating to the VTAM interface.

Profile Modifications

In CS for OS/390 V2R5 and later releases, the PROFILE can be replaced easily without stopping and restarting the TCP/IP stack using the VARY TCPIP,,OBEYFILE (also known as VARY OBEYfile) command. For more information, see the VARY commands in “Chapter 13. Managing the Telnet Server” on page 489. Some rules about PROFILE replacement follow.

- **Complete Replacement**

- New PROFILE statements completely replace the PROFILE statements that were in effect before an update.
- The updates are not cumulative from the previous PROFILE. If only one change is desired in the new PROFILE, the old PROFILE should be updated or copied to another data set member and updated. However, a single port can be modified without including all existing ports in the profile.
- Telnet numbers the profiles sequentially starting with 1.
- The latest profile is also referred to as the CURRent profile.

- **Connection Association**

  New connections use the tables generated by the CURRent profile. The tables generated by older profiles remain active and continue to support any connections that were established when the older profiles were the CURRent profile.

- **Transform Considerations**

  Transform is supported only on a single port. To use transform on a different port, the port using transform must be terminated using VARY TCPIP,,STOP (also known as VARY STOP). Then a VARY TCPIP,,OBEY (or VARYTCPIP,,OBEYFILE) can be used to define transform support to another port.

  Note: The VARY series of commands is sometimes referred to in a short form (for example, VARY OBEY or VARY OBEYFILE), but their full format is more accurate. See "Chapter 4. Configuring the TCP/IP Address Space" on page 97 for more information about the syntax for these commands.

- **Workload Manager Considerations**
  - When the PROFILE is read in by Telnet, the workload manager names are registered.
  - Registration names must be duplicated in later Telnet PROFILES to keep the registration.
  - If a PROFILE causes Telnet to be registered as TN3270 and TNCICS and a later PROFILE update specifies TN3270 only, then the TNCICS name will be deregistered from DNS.

- **PROFILE Table Management**

  When all connections associated with an older PROFILE have ended, the storage for the older PROFILE tables is freed, and the PROFILE is considered inactive. This is permitted because all new connections must use the current PROFILE.

- **PROFILE Processing Rules**

  The PROFILE must include a TELNETPARMS block and a BEGINVTAM block for each port. If both blocks are not present, the PROFILE update does not occur for the port missing the definition blocks. If a single port is used, the stand-alone PORT statement with the INTCLIEN keyword will generate a TELNETPARMS block with all default values. For more information, see "Specifying the Stand-Alone PORT Statement" on page 402.

General Rules for PROFILE Statements

Follow these general rules when making changes to PROFILE statements:

Table 16. General Rules for PROFILE Statements

Parameter	Rules
Case	Names are not case-sensitive and are translated to upper case (except if HFS file KEYRING is specified; this HFS data set name is left in its original case).
Handling of Lists	If one element in an LUGROUP, PRTGROUP, OR IPGROUP is invalid, Telnet flags the invalid element and processes the statement as if the invalid element were not part of the statement. If the statement MUST include at least one element, Telnet will issue a semantic message if the list arrives empty.
Incorrect Input	If you enter incorrect statements or parameters, you see an error message. Telnet ensures that the PROFILE contains at least one of the following before starting processing: 1. TELNETPARMS, ENDTELNETPARMS, BEGINVTAM, and ENDVTAM statements 2. PORT xxx INTCLIEN, BEGINVTAM, and ENDVTAM statements.
IP ADDR	IP addresses can be defined within one group only. • If IP addresses are defined in more than one group, a warning message is issued detailing the IP address that was ignored and the name of the associated group. • No error is listed if an IP address is listed twice in the same group. • If an IP address is listed specifically and also in an IP group, both are accepted and the specific one is used at runtime.
Last versus First	In general, Telnet uses the last valid value or statement (that is, the one with no errors) that was specified. However, if the REPLACEMENT statement does not contain all of the required parameters or the required parameter contains errors in all of its list elements, the statement being replaced will be REMOVED and the REPLACEMENT will not occur. The only exception to the rule is the IPADDRs definition in the IPGROUP statement. In this example: IPGROUP ABC 1.1.1.1. 2.2.2.2 IPGROUP XYZ 2.2.2.2 3.3.3.3 the second IPGROUP statement will receive an error message saying “2.2.2.2 already defined in an IPGROUP, it is ignored.”

Table 16. General Rules for PROFILE Statements (continued)

Parameter	Rules
LU Ranges	You can define the LUs as a range of LUs. Use the following syntax to specify a range of LUs: ►►—*LUbase+LowerRange..LUbase+UpperRange*—►◄ • The symbol "+" represents string concatenation and must ***not*** be coded between the base and the range. • No spaces are allowed within a range definition. • *LUbase* must be the same in both expressions. • *UpperRange* must be greater than the *LowerRange*. • For a numeric range, *LowerRange* and *UpperRange* must both be integers with the same number of digits. Therefore, leading 0's must be specified if required. • For an alphabetic range, *LowerRange* and *UpperRange* must have the same number of letters. *UpperRange* must be greater than *LowerRange* according to the following rule: A < B < C < ... < Z. Every alphabetic character is first converted to uppercase. Special characters #, &, and $ can also be used. • The total length of *LUbase+LowerRange* or *LUbase+UpperRange* must be less than or equal to eight characters. • All LUs in the range must be valid and defined to VTAM for a successful connection. • Each range is limited to 65534 LU names (that is, the number of elements in an LU RANGE must be between 1-65534 inclusively. You can specify multiple ranges to put more than this number into an LUGROUP. • Hexadecimal ranges will not be acknowledged or implemented For example, the expression `TCP00000..TCP01999` will be expanded to each of the LUs in the group `TCP00000, TCP00001, TCP00002, ... , TCP01999`. `TCP0` forms the *LUbase*, `0000` is the *LowerRange* and `1999` is the *UpperRange*. Telnet allows the alphabetic range to be several characters. For example, LU0AA..LU0CG will contact the same LUs as the previous LU0AA..LU0AZ, LU0BA..LU0BZ, LU0CA..LU0CG. If an incorrect range definition is parsed, it will be ignored and a warning message produced.
Printable Characters	Unprintable characters are not allowed in group names or LU names. The first character must be in the range A, B, ..., Y, Z, #, @, or $. If an unprintable character is specified, an error message is issued and the statement is ignored.
Selection Sequence	Specific sequencing rules for VTAM applications and LUs apply when an end user calls the Telnet server or when an LU is selected. For more information, see "Selection Sequence Considerations for VTAM Applications and LUs" on page 486.
Truncation	During configuration, Telnet ensures that names are the appropriate length. If a name is too long, Telnet issues a message and the statement fails.
Naming Convention	The first character is A–Z, #, @, or $. The remaining characters are A–Z, #, @, $, or 0–9.
Variable Names	Do not use the name of a Profile statement as a variable name on that statement.

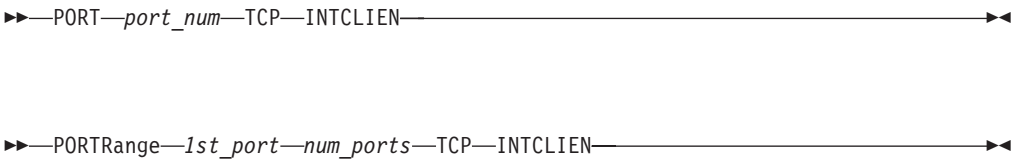
Specifying the Stand-Alone PORT Statement

There are two ways in the PROFILE data set to specify which ports will be used by Telnet. If a single port is used, the stand-alone PORT statement with the INTCLIEN keyword will generate a TELNETPARMS block with all default values. Otherwise, the PROFILE must include a port designation statement under the TELNETPARMS block and a BEGINVTAM block for each port. If both blocks are not present, the PROFILE update does not occur for the port missing the definition blocks.

Stand-Alone PORT and PORTRANGE Statements

The purpose of the stand-alone PORTor PORTRANGE statements with the INTCLIEN keyword is to reserve ports for Telnet use.

Syntax:



Parameters:

port_num
The port number reserved for Telnet use. Port 23 is the default.

TCP
Specifies that TCP is used.

INTCLIEN
The name of the procedure.

1st_port
The starting port of a sequential range of ports to reserve.

num_ports
The number of ports in the sequential range to reserve.

Usage Notes:

- It is not necessary to code either statement to start Telnet. It is advisable to reserve Telnet ports if there is any question of another application taking the ports that Telnet will use.
- The first stand-alone PORT statement with the INTCLIEN keyword will generate a TELNETPARMS block with all default values. The generated TELNETPARMS block is used by Telnet if:
  - Only one port is being used, and
  - No TELNETPARMS block is coded.

  If using multiple Telnet ports, the ports must be defined to Telnet with a TELNETPARMS block. If using one Telnet port, a TELNETPARMS block is recommended.
- If a TELNETPARMS PORT statement is present before or after the stand-alone PORT statement, the TELNETPARMS information will be used. The port numbers should match. If they do not match, the TELNETPARMS port number will be used and the stand-alone PORT statement will reserve a port for no use.

Examples:

```
PORT   23   TCP INTCLIEN    ; Telnet server basic port
PORTR 623 7 TCP INTCLIEN    ; Telnet server secure ports
```

Specifying the TELNETPARMS Statements in the PROFILE Data set

The Telnet server gets some of its configuration parameters through the TELNETPARMS statements. If you want to specify parameters for the Telnet server, update the TELNETPARMS section of the PROFILE data set.

For best performance, use the following parameters in our TELNETPARMS block:

```
TELNETPARMS
      INACTIVE   3600
      TIMEMARK   1200
      SCANINTERVAL 30
   ENDTELNETPARMS
```

To specify TELNETPARMS statements, use the following format in the PROFILE data set:

```
TELNETPARMS
   statements
ENDTELNETPARMS
```

General usage rules for TELNETPARMS statements follow.

- If no statements are entered between TELNETPARMS and ENDTELNETPARMS, then Telnet will use the default values for each of the statements explained below.
- The TELNETPARMS block may be specified anywhere in the profile. If a block is duplicated, the last values are used.
- If there are multiple statements within the TELNETPARMS block, the last statement with no errors is used.
- The ENDTELNETPARMS statement ends the list of TELNETPARMS statements. If this statement is omitted, each parameter that follows is evaluated against the list of valid parameters, and error messages are generated for those that do not match.
- You can update the TELNETPARMS values in a new PROFILE using the VARY TCPIP,,OBEYFILE (also known as VARY OBEYfile) command.

The TELNETPARMS statements follow.

BINARYLINEMODE Statement

The TELNETPARMS BINARYLINEMODE statement is used to prohibit translation of characters between EBCDIC and ASCII during linemode sessions. The default is to allow translation.

Syntax:

Parameters:

BINARYLINEMODE
Allows incoming Telnet linemode session to operate in binary mode (with no translation between EBCDIC and ASCII).

Examples: The BINARYLINEMODE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  BINARYLINEMODE
  :
ENDTELNETPARMS
```

CLIENTAUTH

This optional statement allows you to specify whether or not client authentication is used for the secure Telnet port. Because this function is specified on a per-port basis, each Telnet port can elect to support or not support client authentication.

Syntax:

Parameters:

SSLCERT
Specifies that the SSL handshake process authenticates the client certificate as well as the server certificate. This check verifies that the client has received a certificate from a trusted certificate authority (CA).

SAFCERT
Specifies that the SSL handshake process authenticates the client certificate and also provides additional access control prior to presenting the USSMSG screen through the installation's SAF compliant security product (for example, RACF) as follows:

- Verifies that the client certificate has an associated userid defined to the security product. The certificate must first be defined to the security product to obtain this validation. For more information about adding certificates to RACF, see the description of the RACDCERT command in the *OS/390 Security Server Command Language Reference*.
- For security products that support the 'SERVAUTH' class, installations can also obtain a more granular level of access control. If the installation has activated the SERVAUTH class and provided a profile for the port in the 'SERVAUTH' class, only users specified in the profile are allowed to connect into the port. The security product profile name is specified in the following format:

  ```
  EZB.TN3270.sysname.tcpname.PORTnnnnn
  ```

  where:
  - sysname is the name of the MVS system image.
  - tcpname is the TCP stack name.
  - nnnnn is the port number, with leading zeros specified.

  Note: Wildcards can be used in the profile name if supported by the installation's security product.

  For example, the security product profile name for port 23 running on the TCP stack called TCPCS on system MVS is EZB.TN3270.MVS.TCPCS.PORT00023.

  To protect all ports with a single profile, the following security product profile name could be used: EZB.TN3270.MVS.TCPCS.PORT* (if the installation's security product supports wildcards in profile names).

All security product rules (for example wildcards, PROTECTALL, and so on) apply.

NONE
No client authentication checks are to be done before displaying the USSMSG screen.

Usage Notes:

- If CLIENTAUTH is not coded, CLIENTAUTH NONE is the default. If CLIENTAUTH is specified but is not followed by SSLCERT, SAFCERT or NONE, an EZZ0401I warning message is issued, and SAFCERT is assumed for maximum protection.
- As with all TELNETPARMS parameters, this parameter can be modified by changing the profile and issuing a VARY TCPIP,,OBEYFILE (also known as VARY OBEYfile)with the new profile.
- The CLIENTAUTH parameter indicates the type of client authentication to be done before the Telnet USSMSG screen is displayed.

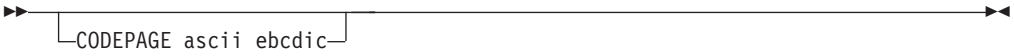
Examples: The CLIENTAUTH statement must be between the TELNETPARMS and ENDTELNETPARMS statements. This parameter is only meaningful for secure ports.

```
TELNETPARMS
...
SECUREPORT 992 KEYRING MVS KEYRING.KDB
CLIENTAUTH SAFCERT
.
.
ENDTELNETPARMS
```

CODEPAGE Statement

The TELNETPARMS CODEPAGE statement lets you specify ASCII-EBCDIC translation tables for linemode connections.

Syntax:



Parameters:

ascii
The ASCII code page name

ebcdic
The EBCDIC code page name

Usage Notes: If there is an error in the syntax, a default codepage of ISO8859-1 will be used for ASCII and the language environment codepage taken from locale information will be used as the EBCDIC codepage. If the EBCDIC codepage is in error, a default codepage of IBM-1047 is used for EBCDIC.

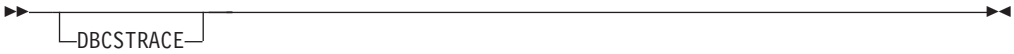
Examples: The CODEPAGE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  CODEPAGE ISO8859-1 IBM-1047
  :
ENDTELNETPARMS
```

DBCSTRACE Statement

The TELNETPARMS DBCSTRACE statement activates tracing within the DBCS load module. The trace records are written to the SYSPRINT file.

Syntax:



Parameters:

DBCSTRACE
Enables DBCS tracing.

Examples: The DBCSTRACE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  DBCSTRACE
  :
ENDTELNETPARMS
```

DBCSTRANSFORM and TRANSFORM Statements

The TELNETPARMS DBCSTRANSFORM and TRANSFORM statements are used to configure your Telnet server for 3270 SBCS or DBCS. You can define which transform model the Telnet server should load at initialization. The default is to load no transform model.

Note: Transform is supported only on a single port. To use transform on a different port, the port using transform must be terminated using VARY TCPIP,,STOP (also known as VARY STOP). Then a VARY TCPIP,,OBEYFILE (also known as VARY OBEYfile) can be used to define transform support to another port.

Syntax:

Parameters:

DBCSTRANSFORM
Specifies that the Telnet server should load the 3270 DBCS transform module, TNDBCSTM, at initialization. The TNDBCSTM module must be in a data set in the system's search list. You can find the module in the installation data set, *hlq*.SEZALINK. If you are using the 3270 DBCS transform mode, the TCP/IP address space may require additional virtual storage.

TRANSFORM
Specifies that the Telnet server should load the 3270 SBCS transform module, TNSIMHPI, at initialization. This module is only available as a third-party product.

Examples: The TRANSFORM or DBCSTRANSFORM statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  DBCSTRANSFORM
  :
ENDTELNETPARMS
```

DISABLESGA Statement

The TELNETPARMS DISABLESGA statement permits the transmission of GO AHEAD by Telnet. It is negotiated by both client and server. Using DISABLESGA increases the overhead for a full duplex terminal and a full duplex connection. The default is to suppress transmission of GO AHEAD.

Syntax:

Parameters:

DISABLESGA
Permits the transmission of GO AHEAD by Telnet.

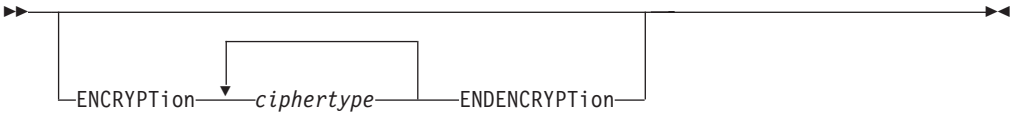
Examples: The DISABLESGA statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  DISABLESGA
  :
ENDTELNETPARMS
```

ENCRYPTION and ENDENCRYPTION Statements

Each of the Telnet FMIDs supports a specific set of encryption algorithms. The ENCRYPTION/ENDENCRYPTION block allows the selection of a subset of the supported algorithms to use for this port. The ENCRYPTION/ENDENCRYPTION block applies only to a Telnet SECUREPORT that serves SSL V3 clients.

Syntax:



Parameters:

ciphertype
The encryption types to use for this port. The order in which the encryption types are specified is not significant.

Usage Notes:

- If this keyword is not coded, the Telnet server supports all encryption methods available for the fmid.
- The server specifies the list of encryption types that it supports. The client controls which of the available ciphers will be used for the data encryption by specifying the desired ciphers in order of preference. The actual cipher used is the best match between what the server supports and what the client requests. If the server does not support any of the ciphers the client requests, the SSL handshake will fail and the connection will be closed.
- Following are the encryption methods that Telnet supports and the FMIDs for which the encryption methods are valid:

	Available (y) in:			
SSL V3 Cipher Type	base FMID	Level1 FMID	Level2 FMID	Level3 FMID
SSL_NULL_Null	y	y	y	y
SSL_NULL_MD5	y	y	y	y
SSL_NULL_SHA	y	y	y	y
SSL_RC4_MD5_EX		y	y	y
SSL_RC4_MD5				y
SSL_RC4_SHA				y
SSL_RC2_MD5_EX		y	y	y
SSL_DES_SHA			y	y
SSL_3DES_SHA				y

Examples:

```
TELNETPARMS
    SECUREPORT 501 KEYRING  HFS /tmp/tcpcs.kyr
    ENCRYPT  SSL_DES_SHA  SSL_3DES_SHA ENDENCRYPT
    .
    .
    .
ENDTELNETPARMS
```

In this example, an HFS keyring file is used. All connections for this port will use SSL security with either DES or triple DES encryption algorithms.

Note: The example above will fail unless either the DES FMID or 3DES FMID feature is installed. If the DESS FMID feature is installed, the profile will give a warning message that SSL_3DES_SHA is not available and all connections will use the SSL_DES_SHA encryption. If the SSL client does not support one of the specified encryption methods, the SSL handshake fails and the connection is closed.

FULLDATATRACE Statement

With the TELNETPARMS FULLDATATRACE statement specified, all data to and from the client and all data to and from VTAM will be completely traced. Without FULLDATATRACE specified, the first 64 bytes of data are traced.

Syntax:

Parameters: None

Examples: The FULLDATATRACE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  FULLDATATRACE
  :
ENDTELNETPARMS
```

INACTIVE Statement

The TELNETPARMS INACTIVE statement defines the terminal inactivity time-out. A session with no client-VTAM activity for the specified time will be dropped.

Syntax:

Parameters:

0 The default inactivity time-out of 0 means no inactivity time-out.

sec
Sets the inactivity time-out to a specified number of seconds. When a connection has been inactive for the specified number of seconds, it is closed. The default inactivity time-out is 0, meaning no inactivity time-out. This number must be an integer in the range 0–99999999.

Examples: The INACTIVE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  INACTIVE 200
  :
ENDTELNETPARMS
```

NOTN3270E Statement

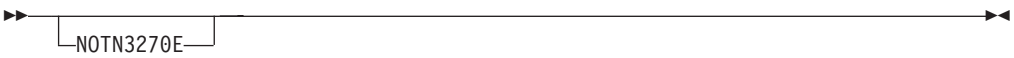
The TELNETPARMS NOTN3270E statement controls the negotiation begun by the server.

- If you specify NOTN3270E, all connection negotiations begin with TN3270 and will never try to connect as TN3270E connections.
- If you do not specify NOTN3270E, all connection negotiations begin with TN3270E and then drop to TN3270 only if the client indicates it can not support TN3270E.

NOTN3270E is useful if there are a significant number of clients that can not tolerate the TN3270E negotiation or if RESTRICTAPPL has a large number of USER statements that also include LU names. TN3270E connections assign the first available LU name during connection negotiation. That may not be the LU name desired based on the APPLID and USERID specified later. TN3270 connections do not choose an LU name until the APPLID and USERID are specified.

If NOTN3270E is used, no TN3270E functions, including printer support and client response, will be available.

Syntax:



Parameters: None

Examples: The NOTN3270E statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  NOTN3270E
  :
ENDTELNETPARMS
```

OLDSOLICITOR Statement

The TELNETPARMS OLDSOLICITOR statement controls the initial cursor placement on the solicitor panel.

- If you specify OLDSOLICITOR, the cursor is placed after:

  ```
  Enter Your Userid:
  ```

- If you do not specify OLDSOLICITOR, the cursor is placed after:

  ```
  Application:
  ```

Syntax:

Parameters: None

Examples: The OLDSOLICITOR statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  OLDSOLICITOR
  :
ENDTELNETPARMS
```

Port Designation Statements

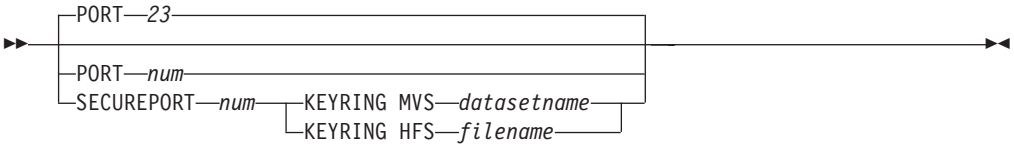
The TELNETPARMS PORT statement defines a basic port to the Telnet server. If only a single port is used, the port can also be set using the PORT statement with the INTCLIEN keyword. For more information, see “Specifying the Stand-Alone PORT Statement” on page 402.

The TELNETPARMS SECUREPORT statement defines which port the Telnet server will listen on for secured connection requests from a client using the SSL protocol. If no TELNETPARMS block with a SECUREPORT statement is found in the profile, this particular Telnet server will not support secured access from a client.

Note: To get the security benefit from a secureport, one of the SSL encryption features must be installed. For more information, see the *OS/390 SecureWay Communications Server: IP Migration*.

Only one port designation statement can appear within a TELNETPARMS block. If more than one is specified, the last one is used.

Syntax:



Parameters:

PORT
Use the PORT keyword to define a basic Telnet port (that is, a Telnet port that is not using SSL encryption). PORT and SECUREPORT are mutually exclusive.

23 The default port over which Telnet accepts incoming requests.

num
A specified port number.

SECUREPORT
Use the SECUREPORT keyword to define a Telnet port that will use SSL (Secure Sockets Layer) encryption protocols. PORT and SECUREPORT are mutually exclusive.

KEYRING
KEYRING is required with SECUREPORT. It defines the file that contains the certificate to be used during the SSL handshake. For additional information, see “Appendix C. Creating a Keyring File for the Telnet Server” on page 1173.

Note: All SECUREPORTs must use the same keyring file.

Examples: The PORT statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  PORT 15
  :
ENDTELNETPARMS
```

PRTINACTIVE Statement

The TELNETPARMS PRTINACTIVE statement defines the printer inactivity time-out. A printer connection with no client-VTAM activity for the specified time will be dropped.

Syntax:

Parameters:

0 The default inactivity time-out of 0 means there is no inactivity time–out.

sec
Sets the inactivity time-out to a specified number of seconds. When a printer connection has been inactive for the specified number of seconds, it is closed. The default inactivity time-out is 0, meaning no inactivity time-out. This number must be an integer in the range 0 to 99999999. If INACTIVE is non-zero, Telnet will round the PRTINACTIVE number up to the next integer multiple of INACTIVE.

Examples: The PRTINACTIVE statement must be between the TELNETPARMS and ENDTELNETPARMS statements. In the example below, Telnet will round PRTINACTIVE up to 600 seconds to be an integer multiple of INACTIVE.

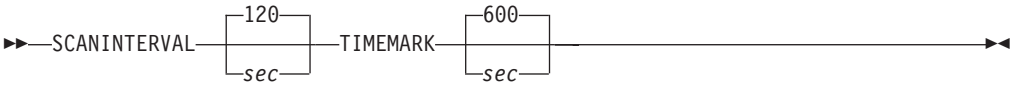
```
TELNETPARMS
  INACTIVE 200
  PRTINACTIVE 500
ENDTELNETPARMS
```

SCANINTERVAL and TIMEMARK Statements

The TELNETPARMS SCANINTERVAL statement defines the interval that Telnet checks connections for inbound TCP/IP activity. It is used in conjunction with the TIMEMARK statement. If the value is specified as 0, or if the statement is not specified, it will be set to the default of 120 seconds (2 minutes). If the SCANINTERVAL is greater than the TIMEMARK value, it will be reset to the TIMEMARK value.

The TELNETPARMS TIMEMARK statement defines the elapsed time the server will use when checking for inbound activity. This value is used to determine whether a connection is considered idle. During SCANINTERVAL processing, if the elapsed time since the last inbound activity to the current time is greater than the TIMEMARK value, the connection is considered idle and a TIMEMARK is sent to the client. The default is 600 seconds (10 minutes). If the connection is still considered idle at the next SCANINTERVAL, that means the client neither responded to the TIMEMARK request nor sent in data. Telnet will drop the connection.

Syntax:



Parameters:

120
The default time between scanning for idle connections or connections waiting to receive a TIMEMARK.

nnn
Changes the SCANINTERVAL time to a specified number of seconds, in the range of 0-16777215

600
The default time after the Telnet server requests a connection before the server closes the connection.

sec
A specified number of seconds, in the range of 0–16777215.

Examples: The SCANINTERVAL and TIMEMARK statements must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  SCANINTERVAL 400
  TIMEMARK 200
  :
ENDTELNETPARMS
```

SINGLEATTN Statement

The TELNETPARMS SINGLEATTN statement causes Telnet to check the data for a PA1 and ATTENTION key combination, x'6CFFEFFFF3', in the data stream sent from the client. If found, Telnet only sends the ATTENTION on to the SNA application x'FFF3'; it discards the PA1. The default is to send both the PA1 and the ATTENTION.

Note: This statement only works in a SNA session. The parameter has no effect in a non-SNA session.

Syntax:

Parameters: None

Examples: The SINGLEATTN statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  SINGLEATTN
  :
ENDTELNETPARMS
```

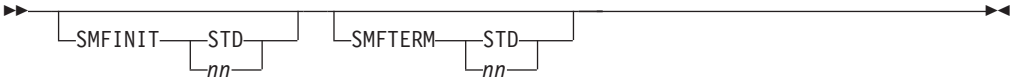
SMFINIT and SMFTERM Statements

The TELNETPARMS SMFINIT and SMFTERM statements are used to configure the Telnet server to write SMF records. This statement specifies the SMF record subtype for LOGON and LOGOFF records.

The TCP/IP server provides services to many clients. Therefore, the system administrator needs access to information about the operation of the clients to resolve any problems and to manage the system. SMF provides logging records for such management purposes. TCP/IP SMF records are independent of the IP connection. They are created for host connections. If no SMF parameters are coded in the TELNETPARMS statement, no SMF records will be written by the Telnet server.

Many products use standard SMF record subtypes. A standard subtype avoids potential double usage and makes it easier for other vendors to write SMF output processing programming and for Telnet administrators to be consistent across multiple machines.

Syntax:



Parameters:

STD
Specifies the standard SMF record subtype for LOGON (20) and LOGOFF (21) records.

nn Specifies the SMF record subtype for LOGON or LOGOFF records. Valid values are integers from 0 to 255. A value of “000” for SMFINIT and SMFTERM indicates that no SMF record will be written for that function.

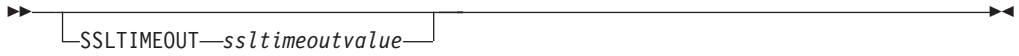
Examples: Following is an example where SMFINIT and SMFTERM are both specified.

```
TELNETPARMS
...
...
SMFINIT STD
SMFTERM STD
...
...
ENDTELNETPARMS
```

SSLTIMEOUT Statement

SSLTIMEOUT provides a unique timeout value for SSL handshake processing. This timeout limits the time SSL handshake processing waits for a client response.

Syntax:



Parameters:

ssltimeoutvalue
Number of seconds in the range 1–86400. Anything outside this range reverts to the default of 5 seconds.

TESTMODE Statement

The TELNETPARMS TESTMODE statement allows an operator to try a PROFILE without having to apply it. In TESTMODE, all the processing and checking are done for an actual update, but at the end of the process, instead of applying the new PROFILE, all data structures are released.

Syntax:

Parameters: None

Usage Notes:

- With the TESTMODE statement, a Telnet administrator can issue an OBEYFILE for a profile and see if there are any syntax or semantic errors without concern for applying an invalid profile. TESTMODE profiles can be processed as often as desired.
- TESTMODE can be included in the initial start-up profile. The end result, however, would be that no port is opened and clients can not connect in. It would be as if no profile statements existed in the initial profiles.

Examples: The TESTMODE statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

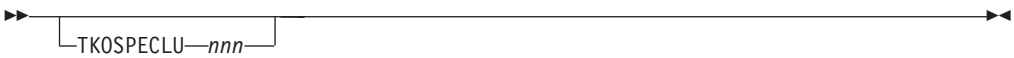
```
TELNETPARMS
  TESTMODE
  :
ENDTELNETPARMS
```

TKOSPECLU Statement

The TKOSPECLU statement allows an operator to specify the period of time, in seconds, the server waits before checking to see if a response was received from the original client. Typically, if a connection request is received that specifies an LU name that is already in use by the server, the server fails the request, and an error code is sent to the client. The TKOSPECLU function suspends a new connection request and sends a DO TIMEMARK command to the original connection that is using the requested LU name. After the specified period of time, the server checks whether there was a response to the DO TIMEMARK command.

If a response or any data has been received by the original connection since the DO TIMEMARK was sent out, the server fails the new connection attempt indicating the LU name is already in use.

Syntax:



Parameters:

nnn Number of seconds the server waits before checking to see if a response was received from the original client.

nnn

Usage Notes:

- You must code a value or the statement is in error.
- If you code zero, the server will always perform the takeover, whether the original session is active or not.
- To take over the session, the new connection must specify the LU name. If administrators want to use this function for a more general purpose, code the @ @LUNAME character substitution in the MSG10 screen so end users will know their LU name if they need to issue a takeover.

TIMEMARK Statement

For a description of the TELNETPARMS TIMEMARK statement, see “SCANINTERVAL and TIMEMARK Statements” on page 420.

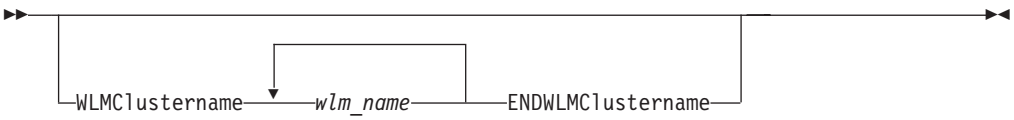
TRANSFORM Statement

For a description of the TELNETPARMS TRANSFORM statement, see “DBCSTRANSFORM and TRANSFORM Statements” on page 410.

WLMCLUSTERNAME Statement

The TELNETPARMS WLMCLUSTERNAME statement is used to register Telnet with the domain name server (DNS) so that the work load manager can be used to balance requests across multiple processors. The work load manager allows the system administrator to balance work load among several machines (and therefore, several VTAMs and Telnet servers) while clients connect using a single name.

Syntax:



Parameters:

WLMClustername
Specifies that the TELNET server is to register in the DNS/WLM Sysplex Connection Balancing Group.

wlm_name
Any legal DNS name up to 18 characters. All servers in a group must register with this name. You can specify an unlimited number of names.

ENDWLMClustername
Required ending to the work load manger name defined with WLMClustername.

Usage Notes:

- The *wlm_name*s are registered with TCP/IP when a profile is read in by Telnet. Registration names must be duplicated in later Telnet profiles to keep the registration.
- If a PROFILE that causes Telnet to be registered as TN3270 and TNCICS is later updated by a second profile to be TN3270 only, the TNCICS name is deregistered from the DNS.
- If the port is QUIESCEd, Telnet will deregister all the names from the DNS. This prevents the DNS from directing additional connections to this Telnet while it is quiesced. Once the port is RESUMEd, all names will again be registered with the DNS.
- If a port is STOPped, Telnet deregisters all of its name from the DNS.
- A single Telnet can register with more than one name. This gives the DNS WLM algorithm more flexibility to direct connections.

Examples: The WLMCLUSTERNAME statement must be between the TELNETPARMS and ENDTELNETPARMS statements.

```
TELNETPARMS
  WLMC
    Groupname1
    Groupname2
  ENDWLMC
  :
ENDTELNETPARMS
```

Specifying BEGINVTAM Statements in the PROFILE Data Set

When you define general characteristics for the Telnet port using the TELNETPARMS statements, you have completed half of the necessary PROFILE configuration task. The other half is completed when you defined LUs, printers, and application characteristics. To do this, you add statements to the BEGINVTAM information block in the PROFILE data set.

The BEGINVTAM statement indicates the start of the list of valid VTAM statements to configure the Telnet server.

The ENDVTAM statement ends the list of VTAM parameters. If this statement is omitted, each parameter that follows is evaluated against the list of valid parameters, and error messages are generated for those that do not match.

Here is an example of how the BEGINVTAM information block looks.

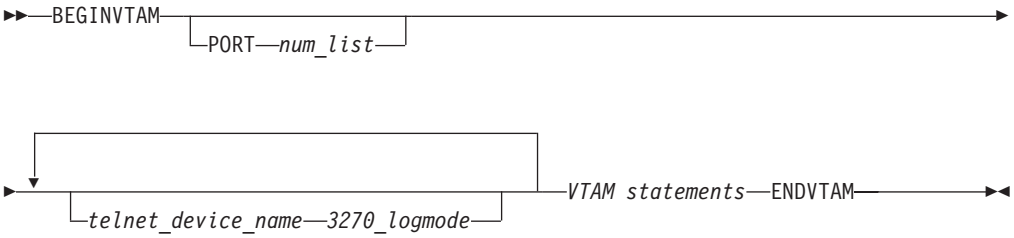
```
BEGINVTAM
    vtam statements
ENDVTAM
```

If more than one Telnet port is used, the first statement in the BEGINVTAM block must be the port designation statement.

The 3270 logmode information can follow the port designation statement. Prior to CS for OS/390 V2R5, the first statement in the BEGINVTAM information block was used to define 3270 logmode. That method for defining logmode is still supported. For more information about this method, refer to “Specifying LOGMODE Using BEGINVTAM” on page 430. The new and preferred method for defining logmodes is to use the TELNETDEVICE statement within the BEGINVTAM block. See“TELNETDEVICE Statement” on page 453 for more information.

Specifying LOGMODE Using BEGINVTAM

Syntax:



Parameters:

PORT *num_list*
: If more than one Telnet port is used, the BEGINVTAM block must define the port or ports that the BEGINVTAM block is associated with. If the BEGINVTAM PORT statement is specified, the 3277 device cannot be the first telnet_device_name because it will be interpreted as a port number. The best solution is to use the TELNETDEVICE statement to define the logmode. For the complete syntax of the PORT statement, see “PORT Statement” on page 447.

telnet_device_name
: The name of the Telnet device. See “TELNETDEVICE Statement” on page 453 for more information on this parameter.

3270_logmode
: The *3270_logmode* entry corresponding to the *telnet_device_name*. Use the defaults in Table 20 on page 486 or user-defined *logmode* entries. If a user-defined *logmode* is used, it must include the same basic bind characteristics as the default. If these device names are not in the configuration data set, the defaults are used. *logmode* entries can refer to either non-SNA or SNA 327x terminals.

The ENDVTAM statement ends the list of VTAM parameters. If this statement is omitted, each parameter that follows is evaluated against the list of valid parameters, and error messages are generated for those that do not match.

Usage Notes: The new and preferred method for defining logmodes is to use the TELNETDEVICE statement within the BEGINVTAM block. See“TELNETDEVICE Statement” on page 453 for more information.

VTAM Statements

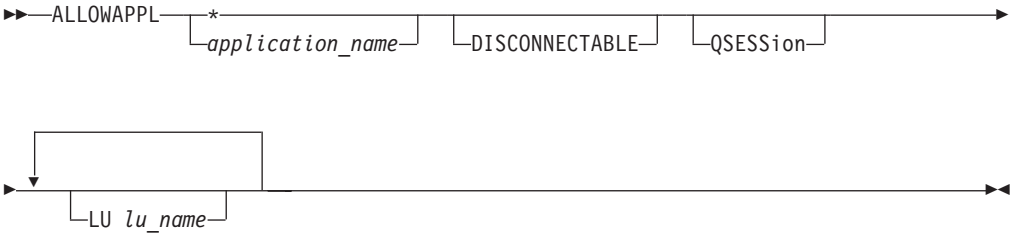
The following statements can appear between BEGINVTAM and ENDVTAM in the PROFILE data set.

ALLOWAPPL Statement

The ALLOWAPPL statement is used to specify the VTAM application names to access. It can be followed by a list of LU names.

Note: The specification of LU names in the ALLOWAPPL is optional. Use of the LUMAP and DEFAULTLUS statements is the preferred method of assigning LUs.

Syntax:



Parameters:

***** A wildcard indicating all applications not previously specified. Specify this statement immediately before the ENDVTAM statement.

application_name
The VTAM application name, as specified in VTAMLST.

DISCONNECTABLE
When DISCONNECTABLE is specified, VTAM notifies the application to disconnect, rather than log off a user, when the session is dropped.

DISCONNECTABLE controls how VTAM terminates a session. If DISCONNECTABLE is specified, the session can be abnormally ended without the user logging off, as when the Telnet QUIT subcommand is issued or the PA1 key is pressed. For more information on the QUIT subcommand, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

QSESSion
Allows operators to select individually which VTAM applications queue their sessions when performing a CLSDST pass. When you use QSESSion to define applications, LUSESSIONPEND is still in effect for all connections. The QUEUESESSION statement is still valid and works as it did previously.

LU lu_name
The logical name of the Telnet LU. This parameter allows you to optionally specify which LUs can establish a session with the named application.

Usage Notes:

- All applications specified in the ALLOWAPPL statement are allowable. You can use % anywhere in the APPL name and * at the end of an APPL name. For example, A%CICS* will allow connections to A1CICS01, A1CICS02, ABCICS4A, and so on.
- When QSESSion is specified, the specified APPL is considered a queue session type of application, and the LU is held when unbinds from passing APPLs are sent to Telnet. Additionally, a table that can store up to 10 application names is

created when a QSESSion application connection is being established. The QSESSion parameter on specific statements does not interfere with LUSESSIONPEND.

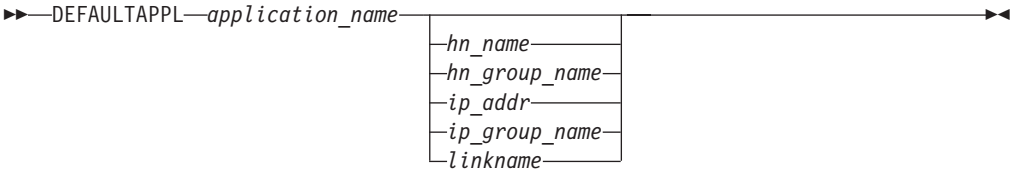
- Applications that do CLSDST Pass also require an ALLOWAPPL statement for the target application.
- Only one appl_name and optional DISCONNECTABLE or QSESSion are allowed per ALLOWAPPL.
- ALLOWAPPL can be followed by a list of LU names. However, LUMAP is the preferred method of locating LUs. If no LU statement is present, an LU specified in an appropriate LUMAP statement is used. If no LU or LUMAP statement is specified, an LU specified in the DEFAULTLUS statement is used. If LU, LUMAP, or DEFAULTLUS statements do not exist or LUs are in use, the connection will fail.

DEFAULTAPPL Statement

The DEFAULTAPPL statement specifies the initial application to which to connect when a Telnet client establishes a connection. The connection is usually a network solicitor or front-end menu system such as SAMON, which allows a user to connect to an application without having to know the actual VTAM name of the application.

Optionally, you can map the application name to either a remote IP address or a network interface (using transparent-mode operation).

Syntax:



Parameters:

application_name
The VTAM application name, as specified in VTAMLST.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

ip_address
The IP address, expressed in dotted decimal form, of a particular client host.

ip_group_name
The group that contains the IP addresses. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

linkname
The uniquely assigned link name. The maximum length is 16 characters. For the SNALINK LU6.2 interface, the maximum length is eight characters.

Usage Notes:

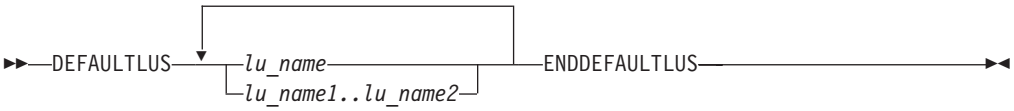
- *hn_name, hn_group_name, ip_address, ip_group_name*, and *linkname* can only match with one application. If more than one application is allocated, only the first application is accepted and the remainder is ignored.
- Any *linkname, hn_group_name,* or *ip_group_name* must first be defined with appropriate LINK, HNGROUP, or IPGROUP statements before being specified in the DEFAULTAPPL statement.
- Previously, DEFAULTAPPL and USSTCP, when coded with the same second parameter (ip_address, ip_group_name, linkname, or blank), were mutually exclusive. Now both are allowed so that in the case of an error, if both are specified, message 7 is displayed (if enabled) even though DEFAULTAPPL is chosen over USSTCP for logon.

- If nothing is specified after the *application_name* parameter, then the statement becomes a global default for all network resources not specified on other statements.
- Hostname specification requires that Telnet be able to resolve a hostname from an IP address by use of the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See Table 2 on page 21 for a description of how TCPIP.DATA is located. Neither the OS/390 environmental variable (Resolver_Config) nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA.

DEFAULTLUS and ENDDEFAULTLUS Statements

The DEFAULTLUS statement starts a list or range of LUs. These LUs are used to represent the terminal when the application selected by the user at that terminal does not meet the criteria for an LU, as defined in the ALLOWAPPL, RESTRICTAPPL, or LUMAP statements. The ENDDEFAULTLUS statement ends the list started by DEFAULTLUS.

Syntax:



Parameters:

lu_name
The name of the LU.

lu_name1..lu_name2
The names of the LUs. When two consecutive dots (..) are included between two LU names in this field expression, a range of LUs will be defined accordingly.

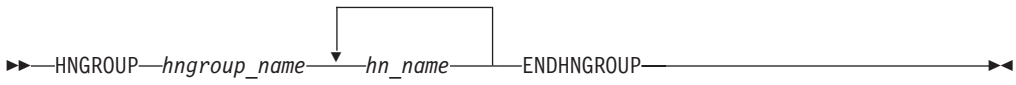
Usage Notes: You can define the LUs in the DEFAULTLUs statement as a range of LUs. For more information about LU names, see “General Rules for PROFILE Statements” on page 400.

HNGROUP and ENDHNGROUP Statements

Use the HNGROUP statement to define a group of host names. These group names can be used on the LUMAP, PRTMAP, DEFAULTAPPL, LINEMODEAPPL, and USSTCP statements when mapping host names.

Note: If you are using host name mapping for filtering, the DNS names of the TN3270 clients will be provided to the CS for OS/390 SNA displays automatically. This occurs automatically because the names must have been resolved for filtering. If you are not using host name mapping for filtering, but want to have TN3270 client names provided to the CS for OS/390 SNA displays, add an HNGROUP name and ENDGROUP name to your TN3270 profile. Choose a name such as A.A within the HNGROUP. The TN3270 client IP address and port is automatically added to the CS for OS/390 SNA displays. The addition of the HNGROUP is only required if you also want the DNS name of the client. If you add the HNGROUP to get DNS name resolution, some delay might occur during connection processing for name resolution. This is optional.

Syntax:



Parameters:

hngroup_name
The group name that contains the network or IP address. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed.

hn_name
An exact, completely qualified host name or a wildcard host name.

Wildcards can be specified in two ways:

- Use a single asterisk (*) to indicate that any value is acceptable for a particular qualifier in a particular position within the host name. For example, *.*.IBM.COM matches USER1.RALEIGH.IBM.COM, but does not match USER1.TCP.RALEIGH.IBM.COM since this name includes an extra qualifier.
- Use a double asterisk (**) to indicate that any number of qualifiers are acceptable to the left of the asterisks. For example, **.IBM.COM matches USER1.IBM.COM, USER1.RALEIGH.IBM.COM, and USER1.TCP.RALEIGH.IBM.COM.

Both wildcard techniques require that the entire qualifier be wildcarded. For example, *USER.IBM.COM is not a valid use of a wildcard. In this case, use *.IBM.COM instead.

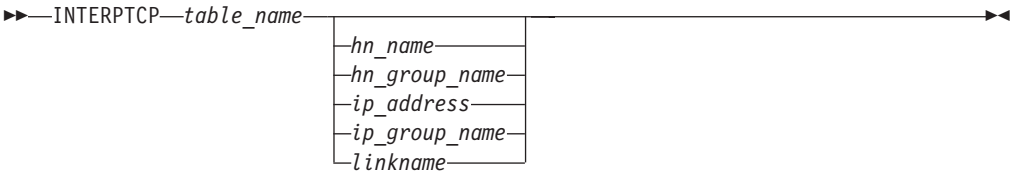
Usage Notes: Hostname specification requires that Telnet be able to resolve a hostname from an IP address by use of the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See Table 2 on page 21 for a description of how TCPIP.DATA is located. Neither the OS/390 environmental variable (Resolver_Config) nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA.

INTERPTCP Statement

The INTERPTCP statement allows you to map a customized interpret table to an IP address, a hostname, or a network interface. This table is used to interpret incoming USS commands before the USS command processor is invoked. If the input string does not match any interpret table entry, the USS command processor parses the input string.

An assembled interpret table load module from VTAM can be used. However, Telnet only matches sequences that use APPLICIDs. Sequences with the USERVAR and ROUTINE options are not matched. If the data string sent to the application should not include the first word, be sure to code the REMOVE=Y option. For coding details, see the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

Syntax:



Parameters:

table_name
The name of the interpret definition table.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

ip_address
The IP address, expressed in dotted decimal form, of a particular client.

ip_group_name
The group name that contains the network or IP address. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z. Special characters #, &, and $ can also be used.

linkname
The uniquely assigned linkname. The maximum length is 16 characters. For the SNALINK LU6.2 interface, the maximum length is 8 characters.

Usage Notes:

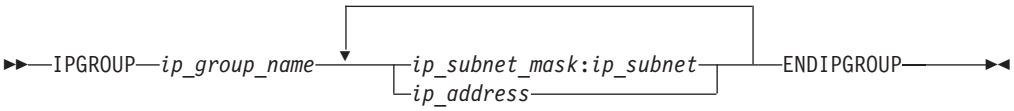
- *hn_name, hn_group_name, ip_address, ip_group_name*, and *linkname* can only match one table definition. If more than one table definition is allocated, the last table is accepted and the remainder is ignored.
- Any *linkname, hn_group_name,* or *ip_group_name* must be defined with the appropriate statements before specifying them in the INTERPTCP statement.

- If nothing is specified after the *table_name* parameter, then the statement becomes a default value for the Telnet client.

IPGROUP and ENDIPGROUP Statements

Use the IPGROUP statement to define a group of IP addresses. These group names can be used on the LUMAP, PRTMAP, DEFAULTAPPL, LINEMODEAPPL, and USSTCP statements when mapping IP addresses.

Syntax:



Parameters:

ip_group_name
The group name that contains the network or IP address. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed.

ip_subnet_mask
A subnet mask, expressed in dotted decimal notation, that identifies which bits of the host field define the network. If the network has no subnets, a subnet mask of zero can be specified.

ip_subnet
The IP address, expressed in dotted decimal form, of the network.

ip_address
The IP address, expressed in dotted decimal form, of a particular host.

Usage Notes:

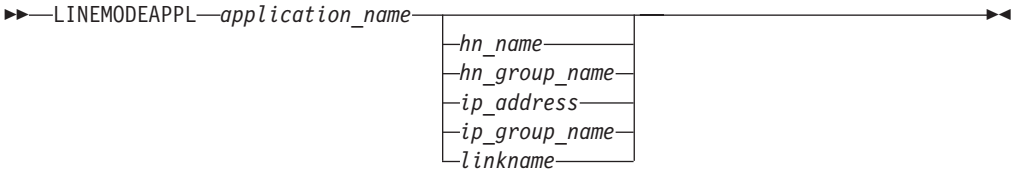
- Any given IP address or combination of IP subnet mask and IP subnet can only appear once within all IP groups.
- The subnet and mask combination has no restrictions, including specific class address specifications.

LINEMODEAPPL Statement

When a Telnet client establishes a line-mode connection, the LINEMODEAPPL statement specifies the initial application to which to connect. The application is usually a network solicitor.

Optionally, you can map the application name to either a remote host name, a remote IP address, or a network interface (using line-mode operation).

Syntax:



Parameters:

application_name
The VTAM application name, as specified in VTAMLST.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

ip_address
The IP address, expressed in dotted decimal form, of a particular host.

ip_group_name
The group that contains the IP addresses. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

linkname
The uniquely assigned link name. The maximum length is 16 characters. For the SNALINK LU6.2 interface, the maximum length is eight characters.

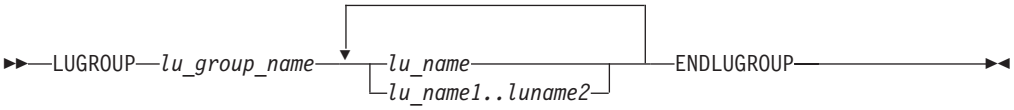
Usage Notes:

- *hn_name, hn_group_name, ip_address, ip_group_name*, and *linkname* can only match one application. If more than one application is allocated, only the first application is accepted and the remainder is ignored.
- Any *linkname, hn_group_name,* or *ip_group_name* must first be defined with appropriate statements before being specified in the LINEMODEAPPL statement.
- If nothing is specified after the *application_name* parameter, then the statement becomes a default value for the VTAM application.
- Hostname specification requires that Telnet be able to resolve a hostname from an IP address by use of the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See Table 2 on page 21 for a description of how TCPIP.DATA is located. Neither the OS/390 environmental variable (Resolver_Config) nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA.

LUGROUP and ENDLUGROUP Statements

Use the LUGROUP statement to define a group of LUs. These group names can be used on the LUMAP statement to represent an LU pool.

Syntax:



Parameters:

lu_group_name
The group name that contains the LU. The maximum length is 8 characters.

lu_name
The name of the LU.

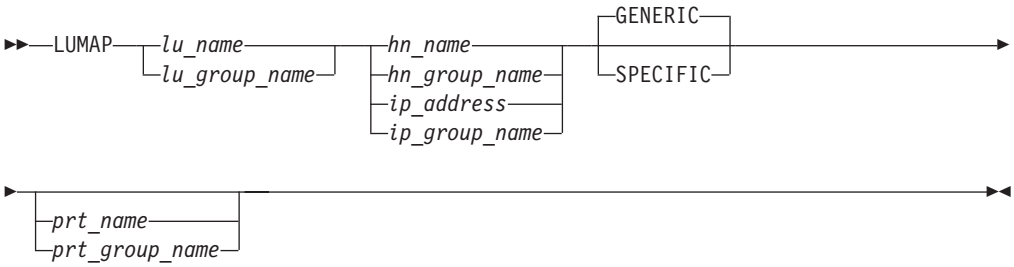
lu_name1..lu_name2
The names of the LUs. When two consecutive dots (..) are included between two LU names in this field expression, a range of LUs will be defined accordingly.

Usage Notes: You can define the LUs in the LUGROUP statement as a range of LUs. For more information about LU names, see “General Rules for PROFILE Statements” on page 400.

LUMAP Statement

Use the LUMAP statement to define the mapping from an LU or group of LUs to an IP address or group of IP addresses.

Syntax:



Parameters:

lu_name
The name of the terminal LU. The maximum length is eight characters. This name ***is*** validated as a VTAM-defined name.

lu_group_name
The group that contains the terminal LUs. The maximum length is eight characters.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

ip_address
The IP address, expressed in dotted decimal form, of a particular host.

ip_group_name
The group that contains the IP terminal addresses.

GENERIC|SPECIFIC
Whichever keyword is listed as the third parameter becomes the keyword. If the third parameter is not SPECIFIC or GENERIC, the default GENERIC is used.

prt_name
The name of a printer LU. The maximum length is eight characters. This name ***is*** validated as a VTAM-defined name at runtime.

prt_group_name
The group that contains the printer LUs. The maximum length is eight characters.

Usage Notes:

- The second parameter (for example, ip_addr, ip_gourp, hostname, or Hn_group) must only appear once in an LUMAP statement in the TCPIP PROFILE. If the second parameter has already been defined in a previous LUMAP statement, then the current LUMAP statement is accepted, and the previous LUMAP statement is ignored.

- Any *lu_group_name, hn_group_name, prt_group_name,* or *ip_group_name* must first be defined with appropriate statements before specifying them in the LUMAP statement.
- When SPECIFIC or GENERIC is found as the third parameter, it is assumed to be the keyword and not a *prt_name* or *prt_group_name*. If the user wants to define a *prt_name* or *prt_group_name* called SPECIFIC or GENERIC, the user MUST specify the SPECIFIC or GENERIC keyword as the third parameter and the *prt_name* or *prt_group_name* as the fourth parameter
- Hostname specification requires that Telnet be able to resolve a hostname from an IP address by use of the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See Table 2 on page 21 for a description of how TCPIP.DATA is located. Neither the OS/390 environmental variable (Resolver_Config) nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA.

LUSESSIONPEND Statement

Use the LUSESSIONPEND statement to allow the user to revert to the DEFAULTAPPL, USSTCP, or Solicitor panel upon termination of the current session. Without this statement, the Telnet server connection is dropped upon termination of the current session.

Syntax:

Parameters: None

Usage Notes:

- If users are presented with a USSTCP screen at logon, when a user logs off a SNA application, the Telnet session will return to the USSTCP screen, from which the user can log onto another SNA application.
- You can specify only LUSESSIONPEND *or* QUEUESESSION; they are mutually exclusive. If both are specified, LUSESSIONPEND is used and a message is issued during TCP/IP initialization.

Examples: The LUSESSIONPEND statement must be between the BEGINVTAM and ENDVTAM statements.

```
BEGINVTAM
  LUSESSIONPEND
  :
ENDVTAM
```

MSG07 Statement

Use the MSG07 statement to activate logon error message processing. Specifying this statement will provide information to the client when a session attempt to the target application fails.

Syntax:

Parameters: None

Examples: The MSG07 statement must be between the BEGINVTAM and ENDVTAM statements.

```
BEGINVTAM
  MSG07
  :
ENDVTAM
```

PORT Statement

If more than one Telnet port is used, the BEGINVTAM block must define the port or ports that the BEGINVTAM block is associated with. The PORT statement must be the first statement in the BEGINVTAM block. If the BEGINVTAM PORT statement is specified, the 3277 device cannot be the first telnet_device_name because it will be interpreted as a port number. The best solution is to use the TELNETDEVICE statement to define logmodes.

Syntax:

Parameters:

num
A specified port number.

num1..num2
A consecutive range of ports starting with *num1* and ending with *num2*. *num2* must be greater than *num1*.

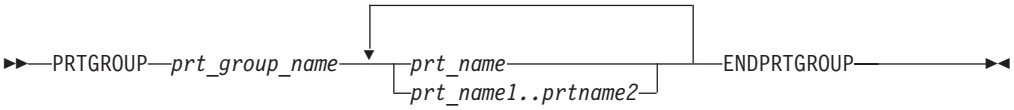
Examples: The PORT statement must be the first statement in the BEGINVTAM block.

```
BEGINVTAM
  PORT 15 23 922
  :
ENDVTAM
```

PRTGROUP Statement

Use the PRTGROUP statement to define a group of printer LUs. These group names can be used on the PRTMAP statement to represent a printer pool.

Syntax:



Parameters:

prt_group_name
The group that contains the printer LU names. The maximum length is eight characters.

prt_name
The name of the printer. The maximum length is eight characters.

prt_name1..prtname2
Name of printers. The maximum length is eight characters. When two consecutive dots (..) are included between two printer names in this field expression, a range of printers will be defined accordingly.

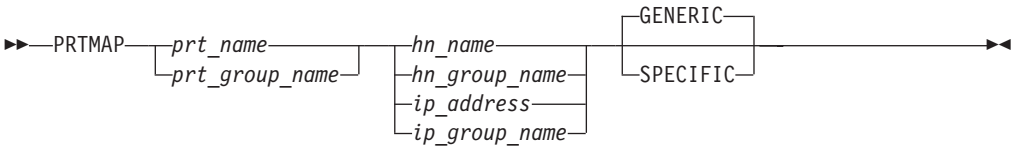
Usage Notes: The printers in the PRTGROUP statement can be defined as a range of printers. For more information about LU names, see “General Rules for PROFILE Statements” on page 400.

Examples: For examples of how the PRTGROUP statement is used, refer to “TN3270E Device Pool Example” on page 482.

PRTMAP Statement

Use the PRTMAP statement to define the mapping from one printer or a group of printers to an IP address or group of IP addresses.

Syntax:



Parameters:

prt_name
The name of the printer. The maximum length is eight characters. This name ***is*** validated as a VTAM-defined name.

prt_group_name
The group that contains the printer LUs. The maximum length is eight characters.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names.

ip_address
The IP address, expressed in dotted decimal form, of a particular host.

ip_group_name
The group that contains the IP addresses.

GENERIC|SPECIFIC
Whichever keyword is listed as the third parameter becomes the keyword. If the third parameter is not SPECIFIC or GENERIC, the default GENERIC is used.

Usage Notes:

- The second parameter must only appear once in an PRTMAP statement in the TCPIP PROFILE. If the second parameter has already been defined in a previous PRTMAP statement, then the current PRTMAP statement is accepted, and the previous PRTMAP statement is ignored.
- Any *prt_group_name, hn_group_name,* or *ip_group_name* must first be defined with appropriate PRTGROUP or IPGROUP statements before specifying them in the PRTMAP statement.
- When SPECIFIC or GENERIC is found as the third parameter, it is assumed to be the keyword and not a *prt_name* or *prt_group_name*. If the user wants to define a *prt_name* or *prt_group_name* called SPECIFIC or GENERIC, the user MUST specify the GENERIC or SPECIFIC keyword as the third parameter and the *prt_name* or *prt_group_name* as the fourth parameter.

Examples: For examples of how the PRTMAP statement is used, refer to "TN3270E Device Pool Example" on page 482.

QUEUESESSION Statement

Use the QUEUESESSION statement to allow the user to revert to the DEFAULTAPPL application upon termination of their Telnet connection. QUEUESESSION is specifically for those applications that support and are required to issue a SIMLOGON type QUEUE macro request. Without this statement, the default would be to terminate the Telnet connection upon termination of their Telnet server session. The DEFAULTAPPL statement must be specified if the QUEUESESSION statement is specified. Using the QSESSion parameter on the ALLOWAPPL and RESTRICTAPPL statements is a better choice for performing this function.

Syntax:

Parameters: None

Usage Notes: You can specify only LUSESSIONPEND *or* QUEUESESSION; they are mutually exclusive. If both are specified, LUSESSIONPEND is used and a message is issued during TCP/IP initialization.

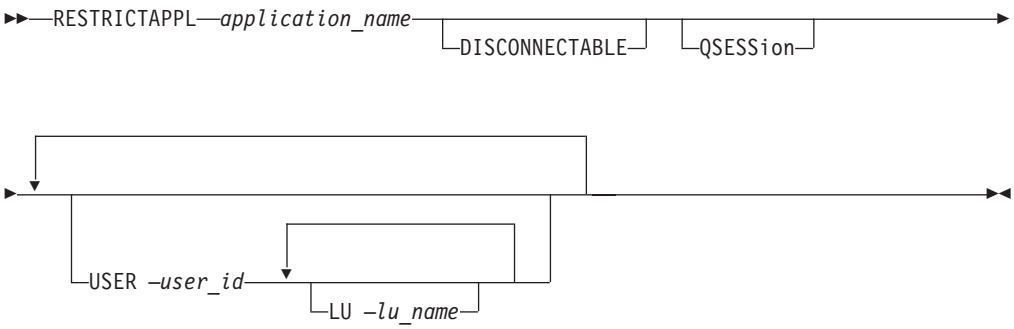
Examples: The QUEUESESSION statement must be between the BEGINVTAM and ENDVTAM statements.

```
BEGINVTAM
  QUEUESESSION
  :
ENDVTAM
```

RESTRICTAPPL Statement

The RESTRICTAPPL statement restricts access to the specified application. This statement must be followed by user parameters defining each user who is authorized to use the application. Users are prompted to identify themselves with a password. RACF or your equivalent security program is used to validate the password. If no user parameters are specified, the application cannot be accessed.

Syntax:



Parameters:

application_name
The VTAM application name, up to eight characters long, as specified in VTAMLST.

DISCONNECTABLE
When DISCONNECTABLE is specified, VTAM notifies the application to disconnect, rather than log off a user, when the session is dropped.

DISCONNECTABLE controls how VTAM terminates a session. If DISCONNECTABLE is specified, the session can be abnormally ended without the user logging off, as when the Telnet Quit subcommand is issued or the PA1 key is pressed.

QSESSion
Allows operators to select individually which VTAM applications queue their sessions when performing a CLSDST pass. When you use QSESSion to define applications, LUSESSIONPEND is still in effect for all connections except those connected to a specific QSESSion application. The QUEUESESSION statement is still valid and works as it did previously.

USER – *user_id*
The user ID, 1–8 characters long.

LU – *LU_name*

The logical name of the VTAM application LU representing the client, up to eight characters long. This parameter allows you to optionally specify which LUs can establish a session with the VTAM host application.

Usage Notes:

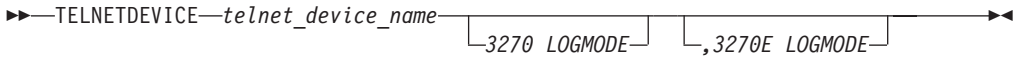
- Only one appl_name and optional DISCONNECTABLE and QSESSION are allowed per RESTRICTAPPL. To handle the second item after the RESTRICTAPPL command, Telnet looks for DISCONNECTABLE, QSESSION, USER, or the start of a new statement.
- Only one user_id is allowed per user. To handle the second item after the USER, Telnet will look for an LU, another USER, or the start of a new statement.
- When QSESSion is specified, the specified APPL is considered a queue session type of application, and the LU is held when unbinds from passing APPLs are sent to Telnet. Additionally, a table that can store up to 10 application names is created when a QSESSion application connection is being established. The QSESSion parameter on specific statements does not interfere with LUSESSIONPEND.
- Applications that do CLSDST Pass also require a RESTRICTAPPL or ALLOWAPPL statement for the target application.
- RESTRICTAPPL user can be followed by a list of LU names. However, LUMAP is the preferred method of locating LUs. If no LU statement is present, an LU specified in an appropriate LUMAP statement is used. If no LU or LUMAP statement is specified, an LU specified in the DEFAULTLUS statement is used. If LU, LUMAP, or DEFAULTLUS statements do not exist or LUs are in use, the application will fail.

TELNETDEVICE Statement

The TELNETDEVICE statement lets you specify a logmode for a device type anywhere within the BEGINVTAM/ENDVTAM block, instead of just at the beginning of the block (as with the BEGINVTAM statement). This statement accepts two logmodes: one for 3270 connections and one for 3270E connections.

In previous releases, logmode was defined using the first statement in the BEGINVTAM information block. That method for defining logmode is still supported. For more information about this method, refer to “Logmode Considerations” on page 485. The new and preferred method for defining 3270 logmodes is to use the TELNETDEVICE statement.

Syntax:



Parameters:

telnet_device_name
The name of the Telnet device. The name is 1-16 characters long. See Table 20 on page 486 for more information on this parameter.

3270 LOGMODE
The *3270 logmode* entry corresponding to the *telnet_device_name*. Use the defaults in Table 20 on page 486 or user-defined *logmode* entries. If a user-defined *logmode* is used, it must include the same basic bind characteristics as the default. *logmode* entries can refer to either non-SNA or SNA 327x terminals.

3270E LOGMODE
The *3270E logmode* entry corresponding to the *telnet_device_name*. Use the defaults in Table 20 on page 486 or user-defined *logmode* entries. If a user-defined *logmode* is used, it must include the same basic bind characteristics as the default. *logmode* entries can refer to either non-SNA or SNA 327x terminals.

Usage Notes:

- Telnet communicates with terminals using either non-SNA (LU0) or SNA (LU2) support or with printers using either SNA character stream (LU1) or 3270 data stream (LU3). The default LOGMODEs in the TCP/IP profile are non-SNA for TN3270 connections. To use a SNA LOGMODE, change the LOGMODE entry to point to an IBM-supplied SNA LOGMODE or create your own. The default logmodes are SNA for TN3270E connections.
- Telnet clients can route SNA print output over TCP/IP. A new class of printer, 328x (device type IBM-TYPE-3287-1), has been added. For more information about printer specifications, see “Logmode Considerations” on page 485.
- Telnet cannot verify that the logmode specified is valid. Problems are detected at runtime.

Examples: Following is an example of the TELNETDEVICE statement.

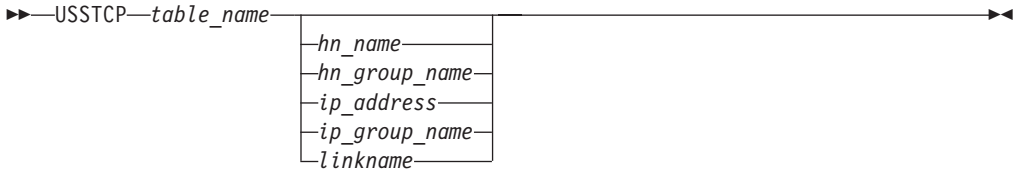
```
BEGINVTAM
   TELNETDEVICE 3278-5-E SNALOG05
   TELNETDEVICE 3278-3-E SNALOG03,SNALOG03
   TELNETDEVICE 3278-2-E ,SNALOG02
    :
 ENDVTAM
```

USSTCP Statement

The USSTCP statement allows you to map a customized USS table to either a remote host name, remote IP address, or a network interface.

You can use an existing table or build a USSMSG table, assemble it and load it into your system library. You can find instructions for this task in “Choosing Telnet Solicitor or USSMSG Logon Panel” on page 460.

Syntax:



Parameters:

table_name
The name of the USS definition table.

Note: Do not code table_name explicitly as USSTCP.

hn_name
The completely qualified host name of a particular client host.

hn_group_name
The group that contains the host names. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z.

ip_address
The IP address, expressed in dotted decimal form, of a particular Telnet.

ip_group_name
The group name that contains the network or IP address. The maximum length of the group name is 16 characters. Nonprintable characters are not allowed. The first character must be in the range A, B, ... , Y, Z. Special characters #, &, and $ can also be used.

linkname
The uniquely assigned linkname. The maximum length is 16 characters. For the SNALINK LU6.2 interface, the maximum length is 8 characters.

Usage Notes:

- *ip_address, hn_name, hn_group_name, ip_group_name*, and *linkname* can only match one table definition. If more than one table definition is allocated, the last table will be accepted and the others will be ignored.
- Any *linkname, hn_group_name,* or *ip_group_name* must be defined with the appropriate statements before specifying them in the USSTCP statement.
- Previously, DEFAULTAPPL and USSTCP, when coded with the same second parameter (ip_address, ip_group_name, linkname, or blank), were mutually exclusive. Now both are allowed so that in the case of an error, if both are specified, message 7 is displayed (if enabled) even though DEFAULTAPPL is chosen over USSTCP for logon.

- If nothing is specified after the *table_name* parameter, then the statement becomes a default value for the Telnet client.
- If you telnet from one MVS to another MVS USSTCP table and then enter an application that is down, message 7 appears if it is enabled. Press the CLEAR key to return to the USSTCP table.
- Hostname specification requires that Telnet be able to resolve a hostname from an IP address by use of the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See Table 2 on page 21 for a description of how TCPIP.DATA is located. Neither the OS/390 environmental variable (Resolver_Config) nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA.

Telnet PROFILE Example

An example of a Telnet PROFILE data set follows:

1. The PORT and SECUREPORT statements define a basic and secure port respectively. Both share the same application parameters.
2. The ALLOWAPPL statement maps a set of LUs that are the only LUs that can be used to connect to an application. In this sample installation, TSO access is only allowed to a subset of LUs:

   ```
   ALLOWAPPL TSO LU LUA LUB LUC
   ```

3. The DEFAULTAPPL statement maps an IP address to a default application to which it automatically will be logged on. In this sample installation, TSO is the default application of IP address 1.1.1.1:

   ```
   DEFAULTAPPL TSO 1.1.1.1
   ```

4. The LUMAP statement maps an IP address to an LU. In this sample installation, the address 1.1.1.1 must be mapped to an LU that is allowed to use TSO. This can be done with the LUMAP statement or the application LU list can be used on its own. If the LUMAP statement is used, it must specify one of the LUs (LUA, LUB, LUC) or an LU group that includes them.

   ```
   LUMAP LUA 1.1.1.1
   ```

Consider the following segment within the PROFILE configuration data set:

```
DEVICE LCS1 LCS BA0
LINK TR1  IBMTR    0 LCS1
LINK ETH1 ETHERNET 1 LCS1

HOME
    1.1.4.8 ETH1
    2.2.4.8 TR1

TELNETPARMS
   PORT 23
ENDTELNETPARMS
TELNETPARMS
   SECUREPORT 992 KEYRING MVS TCPCS.KEYRING
ENDTELNETPARMS

BEGINVTAM
PORT 23 992

LUGROUP sysusrlu CONSLU1..CONSLU7 ENDLUGROUP
LUGROUP
        accntlu  ACCLU01 ACCLU03 ACCLU05 ACCLU07 ACCLU08
                 ACCLU16 ACCLU27 ACCLU38 ACCLU49 ACCLU68
                 ACCLU71 ACCLU75 ACCLU78 ACCLU84 ACCLU95
ENDLUGROUP

IPGROUP
  admin    255.255.240.0:198.25.32.0
ENDIPGROUP

LUMAP accntlu  admin
LUMAP sysusrlu 198.25.32.10
LUMAP CONSLU4 199.28.33.12

DEFAULTAPPL   ADMLOGO   admin
DEFAULTAPPL   CADLOGO   198.25.32.10
DEFAULTAPPL   ANONLOGO  ETH1

LINEMODEAPPL  APCLOGO   admin
LINEMODEAPPL  ANZLOGO   198.25.32.10
LINEMODEAPPL  NONLOGO   ETH1

RESTRICTAPPL  CADLOGO
  USER NIGEL
    LU CONSLU1 LU CONSLU2 LU CONSLU3

ALLOWAPPL ANZLOGO
  LU CONSLU4 LU CONSLU5 LU CONSLU6 LU CONSLU7

USSTCP ABCLOGON TR1
USSTCP CBALOGON 198.25.32.11
USSTCP CBALOGON 199.28.33.12

ENDVTAM
```

Figure 19. Example TCPIP PROFILE Segment

Explanation

1. Selection by IP Address

   Transparent Mode

   Any workstation with an IP address on the IP subnet 198.25.32.0 (except 198.25.32.10) will use the ADMLOGO application by default.

Line Mode

Any workstation with an IP address on the IP subnet 198.25.32.0 (except 198.25.32.10) will use the APCLOGO application by default.

2. Selection by Link Name

   Transparent Mode

   Any users on the ETH1 network interface will use the ANONLOGO application by default if their IP addresses or IP subnets do not match those mapped with ADMLOGO or CADLOGO.

   Line Mode

   Any users on the ETH1 network interface will use the NONLOGO application by default if their IP addresses or IP subnets do not match those mapped with APCLOGO or ANZLOGO.

3. Selection of LU Name

   When any of the workstations on the 198.25.32.0 subnet (except a workstation with the IP address 198.25.32.10) attempts to connect to the MVS system, they will be assigned one of the LUs ACCLU01, ACCLU03, ACCLU05, ... , ACCLU84, ACCLU95. These LUs must be defined to VTAM.

   If the user connects in transparent mode, the application ADMLOGO will be run, and if in line mode, the application APCLOGO will be run.

4. Effect of RESTRICTAPPL and ALLOWAPPL

   Transparent Mode

   When the workstation with an IP address of 198.25.32.10 attempts to connect to the MVS system, it will be assigned one of the LUs CONSLU1, CONSLU2, or CONSLU3 with application CADLOGO, provided that the user is NIGEL and the correct password is entered. The LUs CONSLU1, CONSLU2, and CONSLU3 must be defined to VTAM.

   If no LU names are mapped to the user using the LUMAP statement, selection of LUs will be solely based on the RESTRICTAPPL or ALLOWAPPL statement's associated LUs. Because the LUs CONSLU4, CONSLU5, CONSLU6, and CONSLU7 are not included in the RESTRICTAPPL statement they will not be available to a Telnet session from 198.25.32.10.

   Line Mode

   When the workstation with an IP address of 198.25.32.10 attempts to connect to the MVS system, it will be assigned one of the LUs CONSLU4, CONSLU5, CONSLU6, or CONSLU7 with application ANZLOGO. The LUs CONSLU4, CONSLU5, CONSLU6, and CONSLU7 must be defined to VTAM.

   If no LU names are mapped to the user using the LUMAP statement, selection of LUs will be solely based on the RESTRICTAPPL or ALLOWAPPL statement's associated LUs. Because the LUs CONSLU1, CONSLU2, and CONSLU3 are not included in the ALLOWAPPL statement they will not be available to a Telnet session from 198.25.32.10.

5. USS Message 10 Screen Display

   Transparent Mode

   When the workstation with an IP address of 199.28.33.12 attempts to connect to the MVS system in transparent mode, the USS table CBALOGON will be used to display a message 10 screen accordingly. The user can use the ANZLOGO application with the LU CONSLU4. It is not valid to use applications ADMLOGO, ANONLOGO, APCLOGO, and NONLOGO.

   It is also possible to use the restricted application CADLOGO through the workstation 199.28.33.12, but the system administrator has to add the LU CONSLU4 to the RESTRICTAPPL statement first.

Note that the USS message 10 screen will ***also*** be displayed when the workstation with an IP address 198.25.32.11 attempts to connect to the MVS system, even though the IP address 198.25.32.11 belongs to the IP group `admin` which is mapped to the default application `ADMLOGO`. This is because the IP address is explicitly mapped to a USS definition table in the TCPIP PROFILE.

Line Mode

USS Message 10 screen is not available in line-mode operation.

Configuration Considerations

If you read about and made the changes described in "Configuration Options" on page 395, you have done the necessary configuration for TCP/IP. However, you could do other optional configuration that would improve your Telnet implementation. For example, you could define a custom USSMSG screen for people logging on to your host or balance the work load among multiple processors. This section tells you about the other configuration decisions you could make. It includes:

- "Choosing Telnet Solicitor or USSMSG Logon Panel"

  By default, your clients see the Telnet Solicitor screen when they logon to Telnet. But you can create a customized logon screen using the unformatted system service (USS) Table Message 10 screen as the Telnet logon screen. You could also access existing USS Table Messages by specifying a table member name on the USSTCP statement in the BEGINVTAM information block.
- "Considerations for Using TN3270 Enhanced Support (TN3270E)" on page 480

  The Telnet server now supports TN3270 Enhanced mode (TN3270E). With TN3270E, you have additional printer support, additional 3270 function and SNA protocol support, and new flexibility in designating specific and generic device pools.
- "Logmode Considerations" on page 485

  This section describes which logmode entries correspond to which Telnet device type. Telnet communicates with terminals using either non-SNA (LU0) or SNA (LU2) support and with printers using the SNA character stream (LU1) and the 3270 data stream (LU3). To use the IBM-supplied SNA logmode or to create your own, you can use either the TELNETDEVICE statement or the BEGINVTAM logmode statement. The TELNETDEVICE statement is the new and preferred way. For more information, see "TELNETDEVICE Statement" on page 453.
- "Selection Sequence Considerations for VTAM Applications and LUs" on page 486

  When end users connect to the Telnet server, they see an application or a USS message 10 screen based on a defined sequence of operations. Likewise, when an application name is supplied, an LU is allocated according to the application's status (Restricted, Allowed, Default) and the client's IP address. This section includes the rules used by Telnet to select applications and LUs.

Choosing Telnet Solicitor or USSMSG Logon Panel

Telnet is more often being used as the primary method of connection to the SNA mainframe environment. For ease of migration for your end users, Telnet terminal emulation needs to simulate actual SNA terminals as closely as possible. End users may be accustomed to entering abbreviated logon commands, altering logmodes, or entering user data and may expect the familiar VTAM format for this information.

This kind of logon processing can be emulated in Telnet using the unformatted system service (USS) Message 10 screen as the Telnet logon screen.

Using the Default Telnet Solicitor Logon Panel

Telnet provides the end user with a default solicitor panel if one of the following is true:

- No DEFAULTAPPL or USSTCP statements were specified.
- A RESTRICTAPPL was requested.

When an end user sees a solicitor panel, the cursor is in the default `Application` field. If you specify the OLDSOLICITOR statement in the TELNETPARMS information block, the cursor defaults to the `Enter Your UserID:` field.

Following is an example of the Telnet solicitor panel:

```
Enter Your Userid:
Password:                          New Password:
Application:
```

The `Userid` field on this panel is used in conjunction with the application to determine if access should be accepted, for example, to determine if a RESTRICTAPPL should have a user ID specified. The `Userid` field also enables users to change their passwords on a system security package.

Creating a Custom USSMSG Screen for Telnet

If you do not want to use the default Telnet Solicitor logon panel, you can implement a customized unformatted system services (USS) screen as the Telnet logon screen. Telnet USSMSG tables can be used to send messages to the client and process commands from the client. TELNET USSMSG tables provide you with a way to emulate VTAM USSMSG terminal operator USSMSG processing.

The customized Telnet USSMSG table that you create can be in addition to the standard USS tables used by VTAM. The table must reside in a data set that is in the system's linklist or be in the STEPLIB statement of TCP/IP's startup procedure.

Telnet USSMSG supports the VTAM session-level USS table to define messages and commands. This table includes:

- Commands (such as LOGON) that Telnet can receive from an end user
- Messages that Telnet sends to an end user
- A translation table that is used for character-coded input from the terminal

To implement the USS table logon screen, follow these steps:

1. Build a USS definition table.
2. Choose which messages to implement.
3. Assemble, link, and load the table into a library that is concatenated as a STEBPLIB in the TCP/IP start-up procedure.
4. Define the USS table in the configuration profile.

These steps are described in the following sections.

Step 1: Building a USS Definition Table: To create the table for use with TCP/IP Telnet, follow these steps:

- Assemble a USSTAB table using the USSTAB, USSCMD, USSPARM, USSMSG, and USSEND macroinstructions. The parameters for these macroinstructions are found under “Discussion and Background”.
- Code a USSMSG macro for messages using assembler statements for character translation. Refer to “Customizing Messages” on page 474 for rules about messages. The following general restrictions apply:
  - You must code the commands and the write control characters (WCCs).
  - Only the 3270 data stream is supported. You can find more information about this data string in the IBM 3270 Information Display System publications
- Follow this with a USSEND macro to indicate the end of the table definition.

The background information for performing this procedure is found below under “Discussion and Background”.

The user’s IP address can be displayed in the screen by coding the @@@@@@@@@@IPADDR keyword. The user’s IP address will replace the keyword in the display. The maximum IP address is 15 characters. The user’s foreign port number can be displayed on the screen by coding the @@PRT keyword. The user’s foreign port number will replace the keyword in the display. See Table 17 on page 470 for other character string substitutions.

If you are using HNGROUP statements for LU mapping, the user’s DNS name can also be displayed on the screen. To substitute the user’s DNS name into your USSMSG, place the @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@IPHOSTNAME string into the text. This allows a maximum name of 40 characters. If fewer than 40 characters are used, the name is padded with blanks on the right hand side.

The following example shows a USS definition table for message 10:

```
USST     USSTAB  FORMAT=DYNAMIC
LOGON    USSCMD  CMD=LOGON,REP=LOGON
         USSPARM PARM=APPLID,DEFAULT=TSO
         USSMSG  MSG=10,BUFFER=BUF10
         USSEND
BUF10    DC    AL2(BUF10E-BUF10S)
BUF10S   DC    X'05C2'
         DC    X'1D60'
         DC    C' DEFAULT USSTABLE FOR TESTING TELNET USSTCP'
         DC    X'11C150'
         DC    X'1D40'
         DC    X'13
BUF10E   EQU   *
         END
```

Note: Any application defined in the USS table must be specified as either an ALLOWAPPL or a RESTRICTAPPL in the TCPIP PROFILE. A default application name will not work in the USS message screen because the connection goes directly to session establishment.

Discussion and Background: You can customize Telnet messages, commands, and the translation table by coding your own USS table. For example, you can change messages to provide non-English text, change characteristics of a message, or change the syntax or default values for a command.

Any changes to Telnet messages or commands should be made with supplementary user-defined USS tables. The IBM-supplied USS table should not be changed or removed since it is used to define all commands and messages that are

not defined in a user-written table. The name of the user-defined USS table should be different from the default table. When Telnet requires a message, command, or input translation table that is defined in the USS tables, it searches the USS tables in a specific order.

- **USS Messages**

  To customize USS messages, you need to code the message you wish to redefine. Telnet will first use the user-defined supplementary USS message table.

- **USS Commands**

  A command must be completely defined in a supplementary user-defined USS table. Telnet uses either the user-defined table or the IBM-supplied default USS tables to locate the complete command definition.

After a USS table is coded, it must be assembled and link-edited as a non-executable module into a library concatenated as a STEPLIB in the TCP/IP startup procedure.

To code a supplementary USS table, create a module using the following macroinstructions:

- The **USSTAB** macroinstruction indicates the beginning of a USS table.
- The **USSCMD** macroinstruction defines commands accepted by the Telnet server.
- The **USSPARM** macroinstruction defines an operand or positional parameter that can be specified on a command identified by the USSCMD macroinstruction. It also defines default values for the operand or positional parameter. There can be multiple USSPARM macroinstructions associated with a USSCMD macroinstruction. For each operand (keyword and positional), code a USSPARM macroinstruction.
- The **USSMSG** macroinstruction defines Telnet USSMSG messages.
- The **USSEND** macroinstruction delimits the end of the USS table.

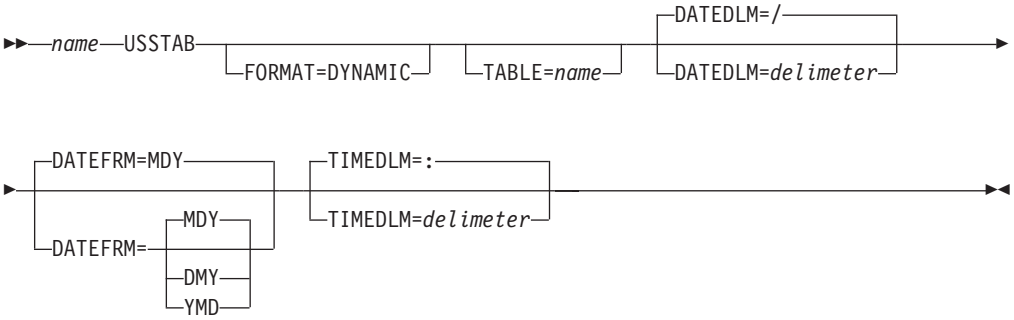
If a user-defined table is coded as part of another module, code an assembler EXTRN definition statement for the table name in that module so the table will be known externally and can be accessed by other modules.

Here are explanations of the five macroinstructions.

USSTAB Macroinstruction:

The USSTAB macroinstruction indicates the beginning of a USS table.

Syntax:



Parameters:

name
Specifies the required CSET name for the USS table.

FORMAT= DYNAMIC
Specifies how the USS table will be formatted. Dynamic is required for Telnet.

TABLE=*name*
Specifies the translation table to be used by Telnet to translate character-coded commands. If a translation table is coded in the specified USS table, the table is used. If no table is coded, the table in the IBM default EZBTPUST is used. If EZBTPUST has been altered and no longer contains a translation table, an internal translation table is used that is the same as the table in EZBTPUST.

DATEDLM
Specifies the character to be used as a delimeter to separate the month, day, and year parts of the date where @@@@DATE is specified in the message. The slash (/) is used if DATEDLM is not specified.

DATEDLM is valid only when FORMAT=DYNAMIC.

Note: An ampersand (&) and single quotation mark (') are not valid delimiters.

DATEFRM
Specifies the date format to be used where @@@@DATE is specified in the message text. Note that the delimiter used between the month, day, and year is specified on the DATEDLM operand.

DATEFRM is valid only when FORMAT=DYNAMIC.

DATEFRM=DMY
Specifies the day, followed by month, followed by year as dd_mm_yy, where an underscore (_) is the delimiter specified on the DATEDLM operand.

DATEFRM=MDY
Specifies the month, followed by day, followed by year as mm_dd_yy, where an underscore (_) is the delimiter specified on the DATEDLM operand.

DATEFRM=YMD

Specifies the year, followed by month, followed by day as yy_mm_dd, where an underscore (_) is the delimiter specified on the DATEDLM operand.

TIMEDLM

Specifies the character to be used as a delimeter to separate the hour, minutes, and seconds parts of the time. The colon (:) is used if TIMEDLM is not specified.

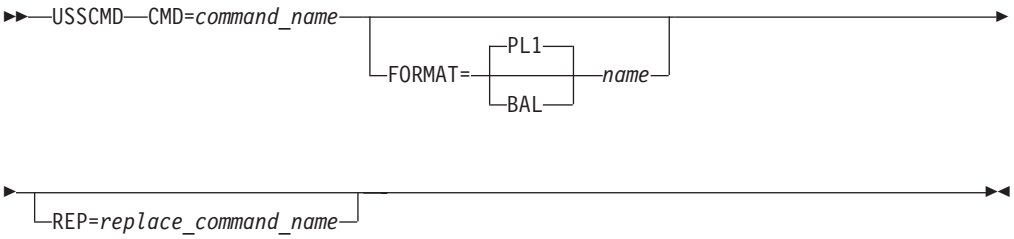
DATEDLM is valid only when FORMAT=DYNAMIC.

Note: An ampersand (&) and single quotation mark (') are not valid delimiters.

USSCMD Macroinstruction:

The USSCMD macroinstruction is used to define a Telnet operator or terminal operator commands.

Syntax:

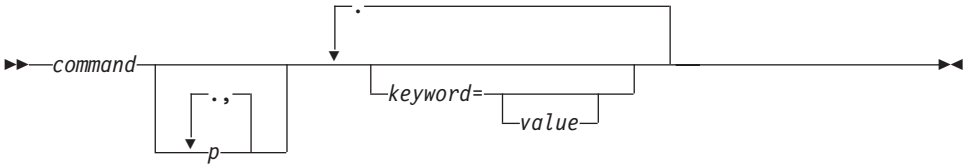


Parameters:

name
Specifies the name assigned to the macroinstruction.

CMD=*command_name*
Specifies the name assigned to the macroinstruction.

FORMAT=BAL
Specifies the user-defined command specified on this USSCMD macroinstruction is in Basic Assembler Language (BAL) syntax.



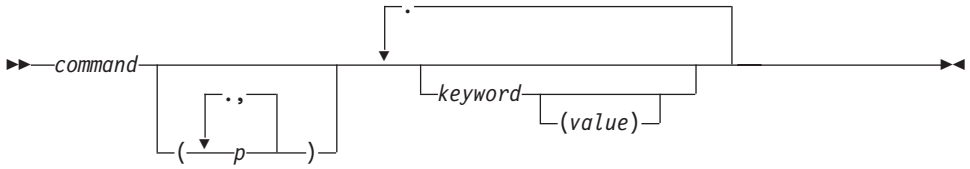
command
identifies the command. It is followed by one or more blanks.

p used to specify one or more positional operand. Positional operands are entered in the format P*n*, where *n* is the position number of the operand. Each operand (unless it is the last in the command) is followed by a comma. Positional operands must appear before any keyword operands.

keyword
used to specify keyword operand associated with the command. Each operand (unless it is the last in a command) is followed by a comma.

value is the value assigned to a keyword operand.

FORMAT=PL1
specifies the user-defined command specified on this USSCMD macroinstruction is in PL/1 programming syntax.



command
identifies the command. It is followed by one or more blanks or by a left parenthesis (that is, positional operands).

p specifies one or more positional operands. Positional operands are entered in the format P*n*, where *n* is the position number. If positional operands are used, the parentheses must be coded.

keyword
is used to enter each operand parameter. Each operand must be followed by one or more blanks or by a value enclosed in parentheses.

value is the value assigned to a keyword operand.

REP=*replace_command_name*
Specifies the valid command that is to replace the user-defined command specified by the CMD operand. If the REP operand is not coded, the value specified in the CMD operator in used.

Usage Notes:

- This Telnet macroinstruction can be coded exactly as the VTAM macroinstruction. If values recognized by VTAM but not by Telnet are present, they are ignored and will not interfere with the execution of this macroinstruction.
- For more information about syntax, input character set, and BAL and PL/1 value restrictions, refer to “Syntax Rules for Terminal Operator Commands” in the VTAM Resource Definition book.
- If the USS table is to be searched twice (REP=VERB coded), this operand is ignored on the second pass of USS processing.

Examples: Here is an example that uses positional and keyword parameters (FORMAT=PL1). The command:

```
LOFF(PROG) L(SNX32705)
```

is issued. Positional and keyword parameters are used with FORMAT=PL1. The following session-level USS table is used:

```
USSTAB
USSCMD CMD=LOFF,REP=LOGOFF,FORMAT=PL1
USSPARM PARM=PL1,REP=APPLID
USSPARM PARM=L,REP=LOGMODE,DEFAULT=SNX32705
USSEND
```

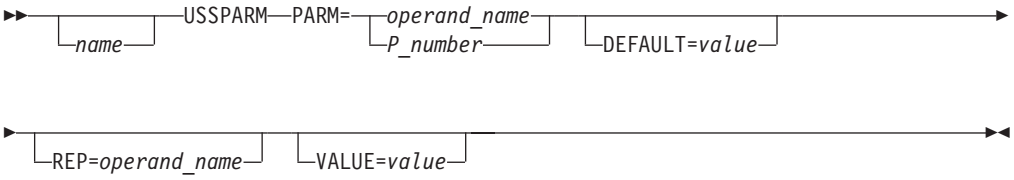
The converted command is:

```
LOFF APPLID(PROG) LOGMODE(SNX32705)
```

USSPARM Macroinstruction:

The USSPARM macroinstruction defines an operand or positional parameter that can be specified on a command identified by the USSCMD macroinstruction. It also defines values for the operand or positional parameter. There can be multiple USSPARM macroinstructions associated with a USSCMD macroinstruction. For each operand (keyword and positional), code a USSPARM macroinstruction.

Syntax:



Parameters:

name
Specifies the name assigned to the macroinstruction.

PARM=*operand_name*
specifies the keyword parameter in the user-entered command to which this USSPARM macroinstruction applies. *operand_name* must be 1–8 alphanumeric characters.

PARM=*P_number*
specifies a positional parameter, where *number* is a decimal integer from 1 to the maximum number of positional parameters for the command. *P_number* indicates the positional parameter in the user-entered command to which this USSPARM macroinstruction applies.

DEFAULT=*value*
specifies a default value to be used if the operand is omitted when the command is entered. If DEFAULT is not specified, the operand is treated as if it were not entered.

If the parameter in the PARM operand allows a network-qualified name to be specified, then the value of DEFAULT can be a network-qualified name.

The DEFAULT and VALUE operands cannot be coded on the same USSPARM macroinstruction. Instead, code two USSPARM macroinstructions with the same value specified for PARM. The macroinstruction specifying VALUE must precede the one containing the DEFAULT operand. If REP is to be specified, it must be on the macroinstruction containing the VALUE operand.

The default value (*value*) must follow the rules for coding assembler DC statements for character (C-type) constants. For example, if you want the following character string for *value*

```
'DUMP'
```

code

```
DEFAULT='DUMP'
```

REP=*operand_name*
specifies the parameter is replaced with *operand_name*. The value for

operand_name must be 1–8 alphanumeric characters. The value of the operand is assigned from the parameter specified by PARM. If PARM specifies an keyword parameter, its value is assigned to the operand specified by REP. If PARM specifies a positional parameter, its value is treated as if it were an operand value and it is assigned to the operand specified by REP.

If REP is not coded, it takes the value of PARM. (That is, the user-entered parameter is used as entered.)

Positional parameters such as P1 and P2 can also be used as operands. For multiple specifications of the same parameter, the last value specified is used (as shown in the examples below).

VALUE=*value*
specifies the default value to be used if the operand specified by the PARM operand is entered without a value.

VALUE is in contrast with the DEFAULT operand, which specifies the default to be used if the operand itself is not entered.

Notes:

1. If multiple VALUE operands are specified for the same operand, the first VALUE operand is used.
2. If the parameter in the PARM operand allows a network-qualified name to be specified, then the value of VALUE can be a network-qualified name.
3. The DEFAULT and VALUE operands cannot be coded on the same USSPARM macroinstruction. To use both operands, code two USSPARM macroinstructions with the same value specified for PARM. The macroinstruction specifying VALUE must precede the one containing the DEFAULT operand. If REP is to be specified, it must be on the macroinstruction containing the VALUE operand. For example,

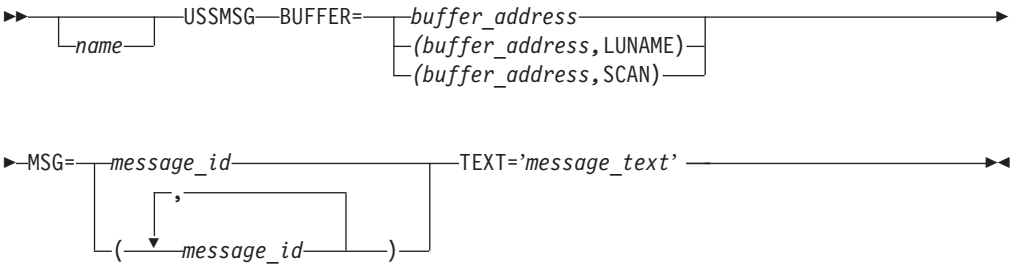
```
USSPARM P=T,REP=TYPE,VALUE=COND
USSPARM P=T,REP=TYPE,DEFAULT=COND
```

Usage Notes: This Telnet macroinstruction can be coded exactly as the VTAM macroinstruction. If values recognized by VTAM but not by Telnet are present, they are ignored and will not interfere with the execution of this macroinstruction.

USSMSG Macroinstruction:

The USSMSG macroinstruction defines Telnet terminal operator messages (USSMSGxx).

Syntax:



Parameters:

name
Specifies the name assigned to the macroinstruction.

BUFFER=buffer_address
Specifies the address (name) of an area of storage defined to contain the message text and a header indicating the length of the message text. The storage area must be formatted as shown in Figure 20.

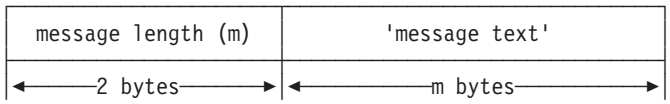


Figure 20. USS Message Layout in Storage

The message text defined in the storage area must follow the rules listed in “Customizing Messages” on page 474.

The message text is sent to the terminal operator as it appears in the storage area. VTAM does not modify or translate the message text (even if FEATUR2=LOWERCSE is specified for the device). You are responsible for including any device-dependent control characters within the message.

BUFFER=LUNAME|SCAN
Specifies that the character strings listed in Table 17, will be replaced with the appropriate values in the position in the message where the character string occurred. The entire string specified by BUFFER will be searched, using the character @.

Table 17. Variables Substituted for USSMSG

Character String	Message Text	Format
@@@@DATE	Current Date	In the format MM/DD/YY
@@@@@@@@@@IPADDR	Client IP Address	15 bytes and leading zeros are not suppressed

Table 17. Variables Substituted for USSMSG (continued)

Character String	Message Text	Format
@IPHOSTNAME	DNS Name of client	Name left justified with trailing blanks if fewer than 40 characters in length
@@LUNAME	Server LU Name (SLU)	8 bytes
@@PRT	Client Port Address	5 bytes and leading zeros are not suppressed
@@@@RUNAME	Failing RU Name	Name is left-justified and trailing blanks are inserted if the name is fewer than 8 characters
@@@SENSE	Server LU Name (SLU)	Name is left-justified and trailing blanks are inserted if the name is fewer than 8 characters
@@@@TIME	Current Time	In the format HH:MM:SS

MSG

Specifies which message or messages will be defined by this macroinstruction.

For terminal operator messages, enter decimal integers in the range 0–14. The numbers 0–14 correspond to the USS messages with message IDs of USSMSG00 through USSMSG14, respectively.

Note: USSMSG00 and USSMSG10 are not defined in the IBM-supplied USS table. If you do not define these messages, no messages are sent for these conditions.

TEXT=*message_text*

specifies the text to replace the USS messages identified by the MSG operand. The message text provided here must follow the rules listed in “Customizing Messages” on page 474.

Note: For a terminal operator message, an error message is produced if you code both the BUFFER and TEXT operands.

On TEXT, you can place any combination of the character strings described in Table 17 on page 470, Telnet will place the strings with the values shown in the table.

Note: Blank suppression always occurs, even if OPT=NOBLKSUP is coded.

Usage Notes: This Telnet macroinstruction can be coded exactly as the VTAM macroinstruction. The OPT and SUPP parameters used by VTAM are not used by Telnet. If they are present, they are ignored and will not interfere with the execution of this macroinstruction.

USSEND Macroinstruction:

The USSEND macroinstruction delimits the end of a USS table.

Syntax:

Parameters:

name
Specifies the name assigned to the macroinstruction

Step 2: Choosing Which Messages to Implement: You can implement any of the message types discussed below.

Listed below are the variables substituted for the USSMSG if variable substitution applies. It is also indicated if the message is not used by Telnet. See the IBM-supplied sample EZBTPUST for an example of these messages.

Table 18. Variables Substituted for USSMSG

Message Number	Variable	Message Text
MSG0	Command	% COMMAND ACCEPTED
MSG1	Command	INVALID % COMMAND SYNTAX
MSG2	Command	% PARAMETER UNRECOGNIZED
MSG3	Command parameter	% PARAMETER EXTRANEOUS
MSG4	1. Command parameter 2. Command parameter value	% PARAMETER VALUE %(2) NOT VALID
MSG5	None	UNSUPPORTED FUNCTION
MSG6	Message Not Used	N/A
MSG7	1. LU Name 2. RU that failed 3. Sense Code information	'%(1) UNABLE TO ESTABLISH SESSION - %(2) FAILED WITH SENSE %(3)
MSG8	None	INSUFFICIENT STORAGE
MSG9	Message Not Used	N/A
MSG10	None	3270 data format screen
MSG11	Message Not Used	N/A
MSG12	None	REQUIRED PARAMETER OMITTED
MSG13	Text after IBMTEST echoed back	IBMECHO %
MSG14	Message number that could not be displayed	USS MESSAGE % NOT DEFINED

Step 3: Assembling, Linking, and Loading the USS Table into Your Library: When you finish building your table, assemble and load it into a load library accessible to the TCP/IP system; for example, one that is reflected in a STEPLIB entry of the TCP/IP start-up procedure.

```
//USSMSG   JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),REGION=1024K,
//         TIME=5
//****************** ==> TPNSUSTB <== ********************************
//ASM     EXEC PGM=ASMA90,PARM='OBJECT,TERM',REGION=1024K
//STEPLIB   DD DISP=SHR,DSNAME=TPO.LINKLIB
//SYSLIB    DD DISP=SHR,DSNAME=VTM410.V4R1M0.SISTMAC1
//          DD DISP=SHR,DSNAME=SYS1.MACLIB
//          DD DISP=SHR,DSNAME=USER42.MACLIB
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(15,1))
//SYSUT2    DD UNIT=SYSDA,SPACE=(CYL,(12,1))
//SYSUT3    DD UNIT=SYSDA,SPACE=(CYL,(12,1))
//SYSPUNCH  DD DISP=SHR,DSNAME=NULLFILE
//SYSLIN    DD DSNAME=&&OBJSET,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(5000,(5000,500))
//SYSTERM   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DSNAME=USER10.USSMSG10.IBMTEST2,DISP=SHR
//LINK    EXEC PGM=IEWL,PARM='LIST,MAP,XREF,RENT,AMODE=31,RMODE=ANY',
```

```
//             REGION=512K,COND=(4,LT)
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD SPACE=(CYL,(15,1)),DISP=(NEW,PASS),UNIT=SYSDA
//SYSLMOD   DD DSNAME=USER10.LINKLIB(IBMTST2),DISP=SHR
//SYSLIN    DD DSNAME=&&OBJSET,DISP=(OLD,DELETE)
```

Step 4: Defining the USS Table in the Configuration Profile: Use the USSTCP statement to define a mapping for IP addresses or link names to default Telnet USSMSG screens. You can specify a single address or a group of IP addresses that have been declared in an IPGROUP statement.

The following example shows how you might code USSMSG10 mappings using USSTCP, IPGROUP, and related statements.

```
HOME
   10.1.4.2      LU0LINK
   10.2.4.2      L62LINK

BEGINVTAM

   IPGROUP IPGRP1 255.255.255.0:138.48.123.0  1.2.1.1 ENDIPGROUP

   USSTCP USSAPC 1.1.1.1
   USSTCP USSCBA LU0LINK
   USSTCP USSABC L62LINK
   USSTCP USSCBA IPGRP1
   USSTCP USSANZ

   DEFAULTAPPL  SAMON  193.1.2.1

   LINEMODEAPPL TSO

ENDVTAM
```

The first USSTCP statement defines a mapping for Telnet connections from IP address 1.1.1.1. Users on this connection will get a USSMSG10 application selection screen from load module USSAPC.

The second and fourth USSTCP statements define Telnet connections that come in from the IPGRP1 IP group or are on the LU0LINK network link. Users on these connections will get a USSMSG10 application selection screen from load module USSCBA.

The third USSTCP statement defines Telnet connections for users on the LU62LINK network link. Users on this connection will get a USSMSG10 application selection screen from load module USSABC.

The fifth USSTCP statement allows all other Telnet connections to get a USSMSG10 application selection screen from load module USSANZ. If this statement was not provided, all Telnet connections not handled by the previous USSTCP mappings would get the 4-line solicitor logon panel.

Customizing Messages: Telnet provides an extensive set of messages that provide information to the end user. You can redefine any of these messages by coding a USSMSG macroinstruction for each message to be redefined. These USSMSG macroinstructions are included in a user-defined USS table.

Using the USSMSG macroinstruction, you can change the message text of any message.

More than one message can be defined using a single USSMSG macroinstruction. If more than one message ID is specified, all of the corresponding messages are changed by the operands specified with this macroinstruction.

When preparing USS messages to be sent to the Telnet client, the following rules apply:

- Single quotation marks in the message text must be specified as in the assembler DC statement for character (C-type) constants.
- The message text must conform to the rules for coding an assembler DC statement for character (C-type) constants. Alternatively, if noncharacter message text is required, the TEXT operand can be specified as a sublist, with each entry coded as the full operand of an assembler DC statement. If the sublist form is used, the assembler does boundary alignment processing.
- You cannot use the 3270 control commands and orders when creating USS messages for SNA devices. For further information, see the component description manual for the appropriate control unit and attached devices.
- When Telnet issues a message, it can supply one or more variable data character strings to be inserted in the message text. The location of the variable data character strings in the message text is indicated by numbered or unnumbered percent signs (%).
  - Numbered percent signs are used when a message has more than one variable character string. The first variable data string is represented by %(1), the second string by %(2), and so on, for as many strings as are provided by Telnet.
  - An unnumbered percent sign is the same as specifying %(1).

  You can rearrange the order of the percent signs in a message to change the order of the variable data in a message.

  Note: Noncharacter data should not contain percent signs (X'6C') because VTAM interprets a percent sign as a point at which to insert variable data into the message.
- Variable text can be used more than once in the same message or can be omitted in the redefined message.
- If the number of a percent sign is greater than the number of variable data character strings provided for that message, that percent sign is replaced by the last available variable data.
- The maximum length of a message after replacement of any percent signs is 251 characters. If a message exceeds 251 characters, the USS table will not assemble. The maximum limit for messages in a message group is 60 characters for each message. Message will be truncated if maximum is exceeded.
- The following characters can be used for USS messages:
  - 26 uppercase and lowercase letters: A–Z, a–z
  - 3 national characters: $ # @
  - 10 numerical digits: 0–9
  - all graphic characters greater than or equal to X'40' (with the exception of X'6C' which is the percent sign).

  National characters (and any graphic or control characters not listed above) are sent to a terminal user only if present in user-specified message replacements specified through the BUFFER operand of the USSMSG macroinstruction.

Note: Because control characters are device-dependent, you should not include them in the data defined by the TEXT or BUFFER operands, unless you can select the appropriate character for the device to which the message will be sent. You may use the percent sign and blanks in data defined by the BUFFER operand. However, do not include the percent sign or blanks in the TEXT operand data because USS treats them differently.

Customizing Commands: Telnet recognizes the following commands that originate from a client:

- "LOGON Command" on page 477
- "LOGOFF Command" on page 478
- "IBMTEST Command" on page 479

The following syntax rules apply to commands you define for terminal operator commands. If these rules are not followed, use an interpret table to convert the character-coded command into a formatted SNA request.

- All graphics characters (greater than or equal to X'40') are allowed.
- Command names and operand values that identify a name can contain 1–8 characters in the following format:

1st character	alphabetical (upper or lowercase) or the national characters @, #, or $
2nd to 8th characters	alphanumeric or the national characters @, #, or $

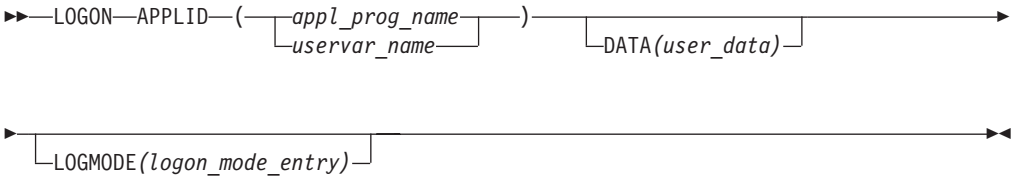
- Command names cannot contain commas except between paired single quotation marks or parentheses.
- Commands names in the BAL format cannot contain blanks or horizontal tab characters except between the command and first operand or between paired single quotation marks.
- For PL/1 format, operand values cannot contain semicolons except between paired single quotation marks Positional operand values cannot contain commas except between paired single quotation marks.
- Operand values cannot contain blanks, horizontal tabs, or unpaired parentheses except between paired single quotation marks.

  Note: There cannot be an odd number of single quotation marks.
- For BAL commands, the terminal operator cannot use the following for operand values:
  - Commas except between paired single quotation marks or parentheses
  - A positional parameter cannot contain equal signs except between paired parentheses or single quotation marks.
- Operand values can contain all graphics characters (greater than or equal to X'40').

LOGON Command:

The LOGON command allows the terminal operator to request a session with an application program.

Syntax:



Parameters:

APPLID
Specifies the name of the application program with which a session is to be established. Code one APPL definition statement for each application program that is in the VTAM domain and include it in an application program node.

DATA
Specifies user data to be made available to the application program's logon exit routine. To embed blanks, code *user_data* as a single-quoted string. The application receiving the user data will not receive the quotes.

LOGMODE
Specifies the logon mode entry that is used to select a set of session parameters for the session to be established.

Usage Notes: The Telnet LOGON command can be coded exactly as the VTAM macroinstruction. The additional VTAM parameters used by VTAM are not used by Telnet. If they are present, they are ignored and will not interfere with the execution of this command.

LOGOFF Command:

The LOGOFF command causes the TCP/IP connection to be dropped between the server and the client.

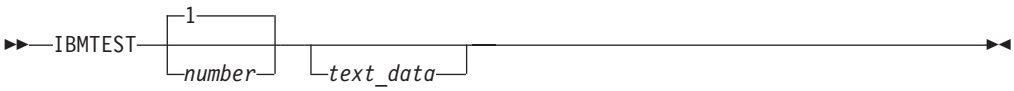
Syntax:

Parameters: The Telnet LOGOFF command can be coded exactly as the VTAM macroinstruction. The additional VTAM parameters used by VTAM are not used by Telnet. If they are present, they are ignored and will not interfere with the execution of this command.

IBMTEST Command:

The IBMTEST command allows the terminal operator to verify that the client is connected to Telnet. Telnet will return the supplied data the specified number of times. Multiple returns may not appear to be multiple because the string is overlaid at the top the of the screen each time. Default values will return the following string once: “ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.”

Syntax:



Parameters:

number
Specifies the number of times the test data should be returned to the terminal. The range is 0-255. The default is 1.

test_data
Specifies the text data to be returned by Telnet. Specify a character string of up to 247 character, or the maximum message length of the terminal, whichever is smaller. If you do not specify a character string, Telnet sends out and returns a string 0-9. You can include blanks as part of this message if the string is surrounded by single quotes.

Usage Notes: The Telnet IBMTEST command can be coded exactly as the VTAM macroinstruction. The additional VTAM parameters used by VTAM are not used by Telnet. If they are present, they are ignored and will not interfere with the execution of this command.

Considerations for Using TN3270 Enhanced Support (TN3270E)

Telnet is most often used as the primary method of connection between client workstations and the SNA mainframe environment. Telnet terminal emulation needs to simulate actual SNA terminals as closely as possible to make this form of remote connection as seamless as possible to end users. RFC1647, also known as TN3270E, adds the ability to simulate specific terminal LU connection and support printer devices and additional SNA functions.

This implementation does not support device name specification and printer support in TN3270 mode as defined in RFC1646.

Here are details about the functionality provided by TN3270E.

- **Device name specification**

  TN3270 clients are assigned a device name (also known as an LU name) based on the client's IP address or the LINKNAME of the device on which the host received the connection. TN3270E clients are assigned a device name using the same algorithm.

  However, a TN3270E client may optionally request that a particular device name be assigned. If the device name is allowed for this client (based on IP address or the LINKNAME of the device on which the connection was received) and available, the client is assigned the requested device name. Otherwise, the request is rejected with an appropriate reason code. In other words, the client can provide the SLU LU name that the server should use. If it is not in the server LU list, the connection will be dropped.

- **328x printer support**

  Telnet clients can route SNA print output over TCP/IP. 328x class printers (device type IBM-3287-1) are now supported. These printers may support both SNA character stream (LU Type 1) and 3270 data stream (LU Type 3) sessions. The 328x printer participates with VTAM applications on the host as a secondary LU and appears to the host application exactly as an actual 3287-class printer, even though it is emulated by software. The server will drop the connection between the Telnet server and the client if a printer request (known by the device type 3287-1) comes in during negotiation on a non-TN3270E connection.

  Telnet 3287 printer emulation has other advantages. It allows the Telnet administrator to use a single product to route SNA print output over TCP/IP. Once printer emulation is available, it is useful to associate printer device names with terminal device names. With association, end users can connect their Telnet terminals to an application and then associate their Telnet printer device with the terminal device.

  For example, a CICS table may specify that if a terminal LU is requesting printer function, the output should be routed to the printer that is associated with that terminal LU. Once this terminal-to-printer association has been set up in the Telnet PROFILE, the end user needs only to remember the terminal LU name when connecting. When the printer session is started at the client, the Telnet client connects using an associated connection specifying the terminal device name. The Telnet server PROFILE ensures that the correct printer device is used.

  For examples of how this association in specified in the Telnet server PROFILE, see "TN3270E Device Pool Example" on page 482. TN3270E does not include support for RFC1646 (use of printers with TN3270).

- **Additional 3270 function support**

  Two new functions are supported: ATTN and SysReq.

The ATTN function in 3270E protocol is sent from the client as an Interrupt Process (IP) to the Server. If the client is connected as a SNA client and sends in an ATTN, the Telnet server forwards a SIGNAL RU to the host application. If the client is connected as a non-SNA terminal, the Telnet server sends a PA1 to the host application. The Telnet server knows the session type based on the bind received from the host application

The SysReq function lets the client enter LOGOFF (in upper, lower or mixed case) after pressing the SysReq key.

- **Additional SNA protocol support**

  Four additional SNA protocol functions are supported by TN3270E:

  – Responses

    The client or host application can request that it receive and provide definite, exception, or no response. Previously, the TN3270 server intercepted these SNA requests from the host and handled them for the client. Now the client will deal with these requests and provide a more realistic interface.

  – Bind images

    The client can request that all binds be forwarded to the client. This allows the client to know more about session presentation, LU type, responses allowed, and so on.

  – SCS-CTL Codes (LU1)

    SCS-CTL is a printer-only negotiated function. It indicates that the client printer will accept SNA Character Stream (SCS) data that is used with LU1 3287 printers.

  – Datastream-CTL (LU3)

    Datastream-CTL is another printer-only negotiated function. It indicates that the client printer will accept standard SNA 3270 data that is used with LU3 3287 printers.

- **Generic and specific device pools for LU names**

  Server LU device name pools (terminal or printer) can be defined as either *generic* or *specific*. Specific pools are used to satisfy requests from clients that request a particular (specific) device name. The client may request to CONNECT to a specific device or ASSOCIATE with a specific device. Generic pool LUs are used to satisfy connection requests where no device name is requested. If a specific request fails to match a device name in the Specific LU pool the Generic pool will be searched. If again no match is found, the connection will be dropped by the server.

  TN3270E recognizes four kinds of device pools: generic and specific terminals (mapped using the LUMAP statement) and generic and specific printers (mapped using the PRTMAP statement). For more information about device pools, see Figure 21 on page 484 and Table 19 on page 484.

  Here are the rules for terminal and printer connections.

  – The client can connect just as it always has and let the server select the terminal LU for the client. This function is also available for printer connections. If no device name is specified, the server selects a name and sends it to the client. If the client agrees, negotiation continues. Otherwise, the connection is dropped.

  – The client can request a specific device name during negotiation. If the server PROFILE definitions allow the use of this name for the particular IP address, that name is reserved for the client and negotiations continue. First the specific group is checked; if no match is found, the generic group is checked. Device name specification can be done for both client terminals and client

printers. The advantage of this is that an end user can connect from anywhere and be assured that he is always connecting to the same LU, either terminal or printer.

– A printer can connect by specifying an active LU terminal name with which it wants to associate. If the server allows the association, the server chooses a print device name and sends it to the printer client. If the client agrees, negotiation continues. Otherwise, the connection is dropped. Printer groups and LU groups are associated using the LUMAP statement.
– If a requested device is already in use, then the request is rejected.
– Printer and terminal pools should have an equal number of device names, or a warning message will be issued. If there are more terminal devices than printer devices, the remaining terminals will not be associated with a printer. If there are more printer device names than terminal devices, the excess printers will not be assigned as associated printers.

For a better understanding of how these rules are put into practice, refer to "TN3270E Device Pool Example".

TN3270E Device Pool Example

Following is an example of how printer and terminal devices are defined in the PROFILE. For this example all clients are in the 9.0.0.0 subnet. An illustration is provided in Figure 21 on page 484.

A specific terminal pool is associated with a specific print pool.

- T1 is associated with P1
- T2 is associated with P2
- T3 is associated with P3

A generic terminal pool is associated with a generic print pool.

- T4 is associated with P4
- T5 is associated with P5
- T6 is associated with P6
- **Defining the Device Pools**

  In this example, the LUGROUP and PRTGROUP statements are being using the define a group of devicenames (LUs) that are used to represent terminals and printers.

```
LUGROUP  LUGRP1  T1 T2 T3  ENDLUGROUP
LUGROUP  LUGRP2  T4 T5 T6  ENDLUGROUP

PRTGROUP  PRTGRP1  P1 P2 P3  ENDPRTGROUP
PRTGROUP  PRTGRP2  P4 P5 P6  ENDPRTGROUP
```

- **Mapping Device Names to an IP Group**

  PRTMAP is used like LUMAP except that it defines a mapping of IP addresses to a PRTGROUP instead of an LUGROUP. Both PRTGROUP and IPGROUP must be defined before they are used on the PRTMAP statement.

```
IPGROUP  IPGRP1  255.0.0.0:9.0.0.0  ENDIPGROUP
LUMAP LUGRP1 IPGRP1
LUMAP LUGRP2 IPGRP1
PRTMAP PRTGRP2 IPGRP1
```

- **Generic and Specific Parameters**

A pool can be either *specific* or *generic*. The PROFILE coding indicates that the defined group is for requested *specific* device names or can be used as a *generic* pool of device names. *Generic* is the default value if neither is specified.

```
LUMAP LUGRP1 IPGRP1 SPECIFIC
LUMAP LUGRP2 IPGRP1 GENERIC
PRTMAP PRTGRP2 IPGRP1
```

- **Mapping a Pool of Printer Device Names to a Pool of Terminal Device Names by Adding PRTGROUP to the LUMAP Statement**

  There must be a one-to-one match between the terminal device pool and the printer device pool. If there is not, a warning message is issued when the PROFILE is processed. If there are more terminal devices than printers, the remaining terminal devices are not associated with any printer. If there are more printer devices than terminals, the remaining printer devices will not be used for associations.

  ```
  LUMAP LUGRP1 IPGRP1 SPECIFIC PRTGRP1
  LUMAP LUGRP2 IPGRP1 GENERIC PRTGRP2
  ```

- **Finishing the PROFILE for TN3270E Support**

  ```
  BEGINVTAM

    LUGROUP LUGRP1 T1 T2 T3 ENDLUGROUP
    LUGROUP LUGRP2 T4 T5 T6 ENDLUGROUP

    PRTGROUP PRTGRP1 P1 P2 P3 ENDPRTGROUP
    PRTGROUP PRTGRP2 P4 P5 P6 ENDPRTGROUP

    IPGROUP IPGRP1 255.5.5.5:9.0.0.0 ENDIPGROUP

    LUMAP LUGRP1 IPGRP1 SPECIFIC PRTGRP1
    LUMAP LUGRP2 IPGRP1 GENERIC PRTGRP2

    PRTMAP PRTGRP2 IPGRP1
    ALLOWAPPL *

  ENDVTAM
  ```

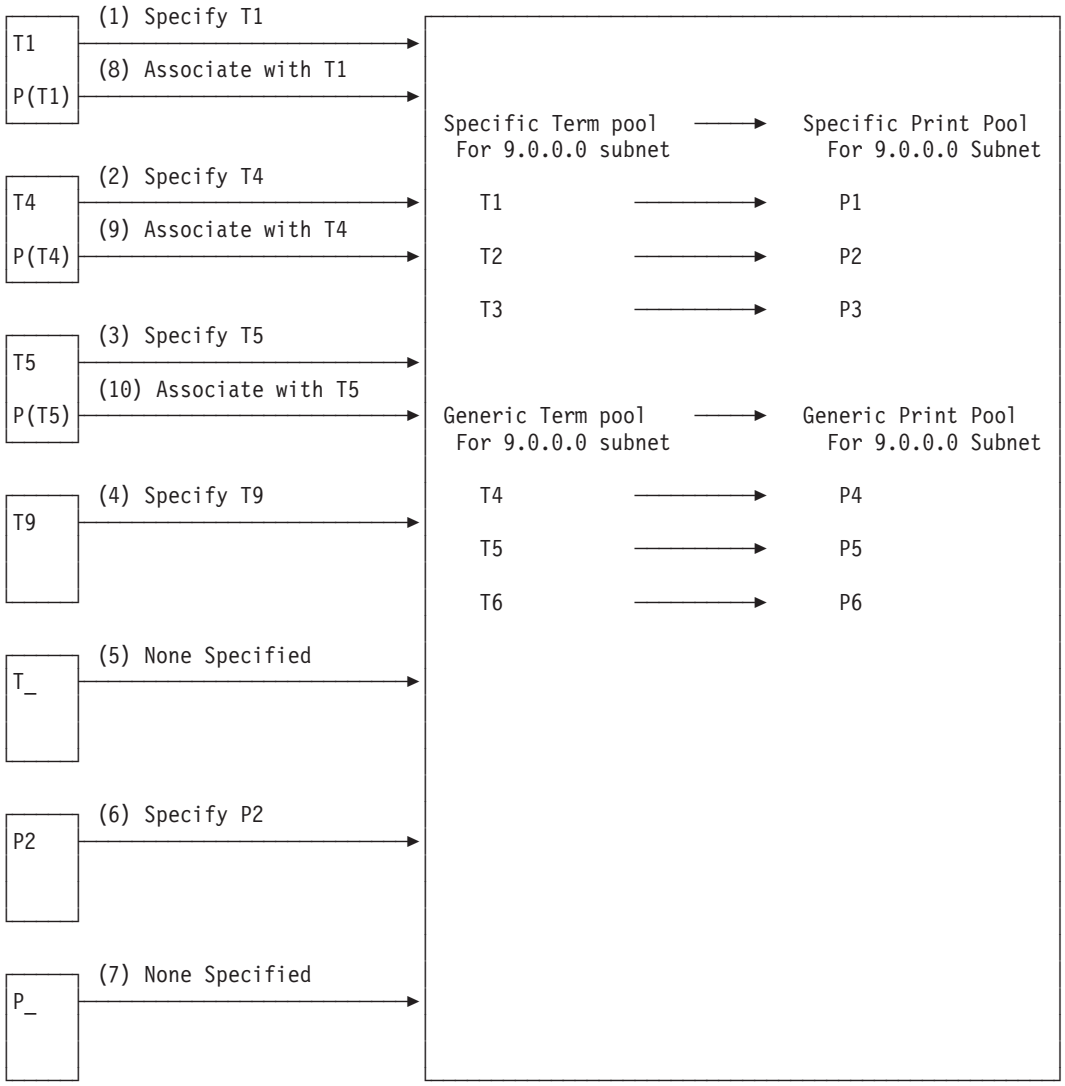


Figure 21. Generic and Specific Device Pool Example

Table 19. Explanation of Generic and Specific Device Pools

Client Action	Result
(1) Specifies T1	• Maps to both specific and generic device pools • Tries specific pool first • Finds match with T1 in specific pool
(2) Specifies T4	• Maps to both specific and generic pools • Tries specific pool first • Doesn't find match so tries generic pool • Find match with T3 in generic pool
(3) Specifies T5	• Maps to both specific and generic pools • Tries specific pool first • Doesn't find match so tries generic pool • Finds match with T5 in generic pool
(4) Specifies T9	• Maps to both specific and generic pools • Tries specific pool first • Doesn't find match so tries generic pool • Doesn't find match so fails connection

Table 19. Explanation of Generic and Specific Device Pools (continued)

Client Action	Result
(5) No device name specified	• Maps to both specific and generic pools • Tries only generic pool and finds no device name specified • Doesn't find match so tries generic pool • Finds T6 available and uses it
(6) Specifies P2	• Maps to both specific and generic pools • Finds match with P2 in specific pool
(7) No device name specified	• Maps to both specific and generic pools • Tries only generic pool and finds no device name specified • Finds match with P4 in generic pool
(8) Specifies association with T1	• Telnet finds T1 active in specific terminal group • Associates terminal group with printer group • Finds associated printer P1 available and uses it
(9) Specifies association with T4	• Telnet finds T4 active in generic terminal group • Associates terminal group with printer group • Finds associated printer P4 unavailable and fails connection
(10) Specifies association with T5	• Telnet finds T5 active in generic terminal group • Associates terminal group with printer group • Finds associated printer P5 available and uses it

Potential TN3270E Problem

Because of 3270E implementation, device name selection is now required during session negotiation. The device name will be chosen before the application is chosen. Working TN3270 connections could result in a mismatch with TN3270E. Consider the following example:

```
BEGINVTAM

  DEFAULTLUS T1 T2 T3 T4 T5 ENDDEFALTLUS
  RESTRICTAPPL APPL1 USER USER54 LU T3
  ALLOWAPPL    APPL2 LU T4

ENDVTAM
```

Two TN3270 connections are started.

- Two solicitor screens appear.
- Specify APPL1, USER65, and a password. The server selects T3.
- Specify APPL2. The server selects T4.

Two TN3270E connections are started.

- Two solicitor screens appear. The server selects T1 and T2.
- Specify APPL1, USER65, and a password. The server fails the connection because of an LU mismatch.
- Specify APPL2. The server fails the connection because of an LU mismatch.

Logmode Considerations

Telnet communicates with either non-SNA (LU0) or SNA (LU2) support. The default logmodes in the TCP/IP PROFILE are non-SNA for TN3270 connections. To use a SNA logmode, you need to change the logmode entry in the BEGINVTAM information block to point to an IBM-supplied SNA logmode or create your own. The default logmodes are SNA for TN3270E connections.

The preferred way to define a LOGMODE is to use the TELNETDEVICE statement that can be issued from anywhere in the BEGINVTAM information block. For more information, see “TELNETDEVICE Statement” on page 453.

Telnet Device Name Parameters

Table 20 lists the Telnet device types and their corresponding default *logmode* entry.

Note: All SNXxxxxx and INTERACT logmodes are SNA logmodes. The rest are non-SNA.

Table 20. Device Type and Logmode Table

Telnet Device Type	3270 Logmode Entry	3270E Logmode Entry
3277	D4B32782	N/A
3278-2-E	NSX32702	SNX32702
3278-2	D4B32782	SNX32702
3278-3-E	NSX32702	SNX32703
3278-3	D4B32783	SNX32703
3278-4-E	NSX32702	SNX32704
3278-4	D4B32784	SNX32704
3278-5-E	NSX32702	SNX32705
3278-5	D4B32785	SNX32705
3279-2-E	NSX32702	N/A
3279-2	D4B32782	N/A
3279-3-E	NSX32702	N/A
3279-3	D4B32783	N/A
3279-4-E	NSX32702	N/A
3279-4	D4B32784	N/A
3279-5-E	NSX32702	N/A
3279-5	D4B32785	N/A
LINEMODE	INTERACT	N/A
DYNAMIC	N/A	SNX32702
3287-1	N/A	SNX32702

Selection Sequence Considerations for VTAM Applications and LUs

When an end user connects to the Telnet server, it selects an application or a USS message 10 screen based on the following sequence of operations:

1. Tries to find a default application mapping according to the client’s host name. If it is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
2. Tries to find a default application mapping according to the client’s IP address. A mapping for the fully specified IP address is checked (for example, 1.1.1.1). If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.

3. Tries to find a default application mapping for an HN group that contains the client's host name. If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
4. Tries to find a default application mapping for an IP group that contains the client's IP address. If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
5. Tries to find a default application wildcard host name mapping according to the client's host name. If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
6. Tries to find a default application mapping according to the client's IP subnet mask and IP subnet, which are specified by the subnet in *mask*:*value* format (for example, 255.255.255:0.1.1.1.0). If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
7. Tries to find a default application mapping according to the client's network interface (link name). If the server is unable to find an application, it tries to find a USS message 10 definition table on the same basis.
8. Tries to find an application defined with a DEFAULTAPPL/LINEMODEAPPL statement that has no optional parameter specified. If the server is unable to find one in transparent mode operation, it tries to find a USS message 10 definition table defined with a USSTCP statement that has no optional parameter specified.
9. If all the previous attempts fail, the solicitor logon panel is displayed to get an application from the user.

When an application name is supplied, an LU is allocated according to the application's status (Restricted, Allowed, Default) and the client's IP address. TCP/IP checks to see if the application is defined as "restricted", "allowed", or "default" (in this order) and allocates an LU, taking into consideration the LUs mapped to the IP address on the LUMAP statement:

If LU names are specified for the application and LU names are mapped to the user's IP address, the LU selected must be in both of these LU lists.

- If one of these LU name lists is not specified, then the LU is selected from the specified list.
- If neither of these LU lists are specified then one of the default LU names is selected.

  Note: LU names cannot be specified for a default application, so only mapped or default LU names are selected.

If TCP/IP cannot successfully allocate an LU following this sequence, the end user receives the Telnet solicitor.

Chapter 13. Managing the Telnet Server

Many networking products (such as VTAM) use VARY commands to change the state of a device, MODIFY commands to modify a value or system parameter, and DISPLAY commands to show information. The Telnet server uses VARY and DISPLAY commands that help you monitor Telnet function and debug problems quickly.

The following set of commands help manage the Telnet server:

- Telnet VARY commands give the operator control over stopping and starting Telnet and allowing clients to connect. These commands include:
  - QUIESCE the port to block any new connections from starting but allow existing connections to continue activity
  - RESUME to end the QUIESCEd state
  - STOP Telnet to end connections to a Telnet port and close the port
  - Start or restart a port using the VARY OBEYFILE command (to update the Telnet PROFILE). Using this command, you can stop Telnet activity on one port and begin activity on a new port without stopping the TCP/IP stack. You can also start activity on new ports without stopping activity on existing ports.
  - ACTivate and INACTivate LUs from the Telnet server's perspective. If an LU is already in use, the INACT command fails. These commands have no effect on the VTAM state of the LU.

  Note: QUIESCE and STOP also de-register the Telnet server from WLM if it was previously registered. This will result in 'host unknown' type messages for clients that attempt to connect to the Telnet server using the WLMCLUSTERNAME in a DNS/WLM sysplex. RESUME will re-register the Telnet server with WLM.
- Telnet DISPLAY commands support multiple dynamic profiles in Telnet. The DISPLAY command set includes these categories:
  - Profile
  - Connection
  - Port
  - Server

Operating the Telnet Server Using the VARY TCPIP Command Set

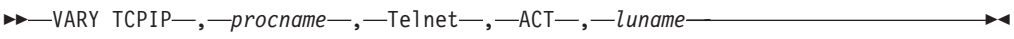
The VARY TCPIP commands give the operator complete control over stopping and starting Telnet and allowing clients to connect. Using the VARY TCPIP commands, you can control the Telnet port and the LUs in the profile table. The combination of the STOP, QUIESCE, RESUME, and OBEYFILE commands gives the operator complete control over when to stop and start Telnet and when to allow end users to connect. To help manage commands related to multiple ports, VARY and DISPLAY commands for the profile, connection, and port categories support a PORT keyword.

Note: Parameters for the VARY TCPIP commands are positional and must be entered in the order shown in the syntax diagrams.

VARY ACT Command

The VARY ACT command changes the availability status of a VTAM LU for Telnet Server usage. ACT enables the specified LU to be a candidate to represent a Telnet client.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

Telnet
Directs the command to the Telnet component.

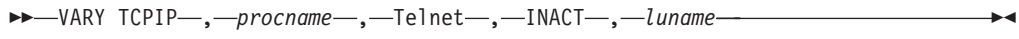
ACT
The activate keyword.

luname
The name of the LU you are activating.

VARY INACT Command

The VARY INACT command changes the availability status of a VTAM LU for Telnet Server usage. INACT disables the LU as a candidate to represent a Telnet client.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

Telnet
Directs the command to the Telnet component.

INACT
The inactivate keyword.

luname
The name of the LU you are deactivating.

Usage Notes

- If the specified LU has an active VTAM session, it will not be affected by this command.
- The VTAM VARY NET,INACT command should be used to end the SNA LU session.
- The TCP/IP VARY DROP command should be used to end the TCP/IP connection.

VARY OBEYFILE Command

The VARY OBEYFILE command is not strictly a Telnet command, but it is used to update the Telnet profile or to restart a Telnet port if the STOP command has been issued. When the VARY OBEYFILE command is issued, the existing profile for a changed port becomes non-current, and a new profile is used for all subsequent connections. The replaced profile continues to service connections that were started when that profile was current. When all connections attached through the replaced profile have ended, the related storage is freed.

Update profiles do not need to contain statements for all active ports. If the profile does not contain a TELNETPARMS block or a BEGINVTAM block for a port that is currently active, the port remains active with its current profile.

As with other profile changes, if the VARY OBEYFILE command changes the port type (for example, from PORT to SECUREPORT or SECUREPORT to PORT), only new connections are affected. Existing connections will continue to use the security type in effect at the time of the connection.

All secure ports must use the same keyring file. To change the keyring file, all ports currently defined as SECUREPORT ports must be stopped.

Syntax

Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

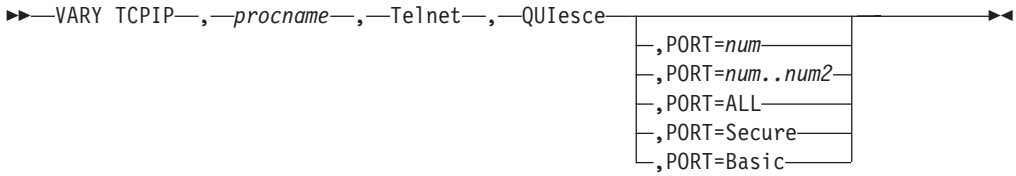
Obeyfile
The Obeyfile command keyword.

dsname
The data set name of the profile that replaces the existing profile.

VARY QUIESCE Command

The VARY QUIESCE command causes the specified port to not accept any new Telnet client connections. Currently established connections continue to be serviced. The current TCP/IP Profile is retained as long as an OBEYFILE command does not create a new profile.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

Telnet
Directs the command to the Telnet component.

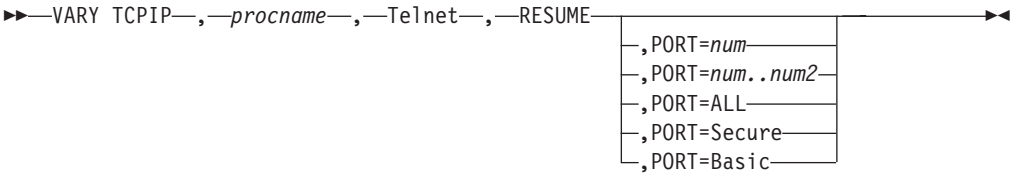
QUIesce
The QUIesce command keyword.

PORT
Specifies the port number, port number range, all active ports, all secure ports, or all basic ports. Using PORT=BASIC selects all ports defined as unencrypted (that is, those whose TELNETPARMS blocks contain an explicit or implicit PORT nnnn statement). Using PORT=SECURE selects ports defined as SECUREPORT. If only one port is active and PORT is not specified, the command affects that one port. Otherwise, PORT is required.

VARY RESUME Command

The VARY RESUME command causes the currently QUIESCEd port to begin accepting new Telnet client connections again using either the existing profile or a new profile. If a VARY OBEYFILE command has been issued while the port was QUIESCEd, a new profile will be used.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

Telnet
Directs the command to the Telnet component.

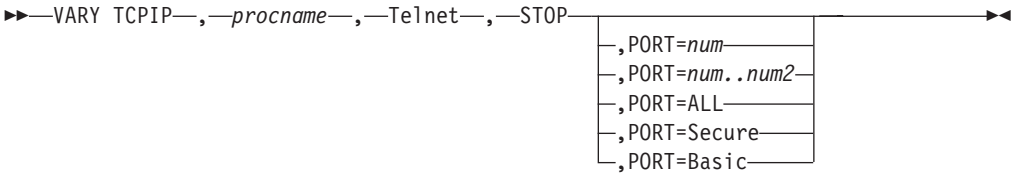
RESUME
The RESUME keyword.

PORT
Specifies the port number, port number range, all active ports, all secure ports, or all basic ports. Using PORT=BASIC selects all ports defined as unencrypted (that is, those whose TELNETPARMS blocks contain an explicit or implicit PORT nnnn statement). Using PORT=SECURE selects ports defined as SECUREPORT. If only one port is active and PORT is not specified, the command affects that one port. Otherwise, PORT is required.

VARY STOP Command

The VARY STOP command ends the port connection and all active connections. STOP does not end all of Telnet. The command processor is still active. You can use a VARY OBEYFILE command to ACTIVATE a Telnet port using the Telnet configuration parameters.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the TCPIP address space.

Telnet
Directs the command to the Telnet component.

STOP
The STOP command keyword.

PORT
Specifies the port number, port number range, all active ports, all secure ports, or all basic ports. Using PORT=BASIC selects all ports defined as unencrypted (that is, those whose TELNETPARMS blocks contain an explicit or implicit PORT nnnn statement). Using PORT=SECURE selects ports defined as SECUREPORT. If only one port is active and PORT is not specified, the command affects that one port. Otherwise, PORT is required.

Operating the Telnet Server Using the DISPLAY Command Set

Telnet DISPLAY commands support multiple dynamic profiles in Telnet. The DISPLAY command set includes these categories:

- Profile
- Connection
- Port
- Server

Note: Parameters for the DISPLAY commands are positional and must be entered in the order shown in the syntax diagrams.

Interpreting DISPLAY Command Output

In the output from some DISPLAY commands, these command abbreviations might appear.

- **Profile Flags**, indicating what general options for the profile are in effect. Possible profiles flags are:
  - B—Binary linemode specified
  - D—Disable SGA specified
  - L—LUSESSIONPEND specified
  - Q—QSESSIONPEND SPECIFIED
  - M—MSG07 specified
  - T/D—
    - TRANSFORM (SimWare) specified
    - DBCS specified
  - S—SINGLEATTN specified
  - O—OLDSOLICITOR specified
  - W—WLM names specified
  - F—FULLDATATRACE specified
  - H—HOSTNAMES obtained
  - N—NOTN3270E specified
- **Connection Option Flags**, indicating what options were negotiated ON with the client. Possible connection option flags are:
  - E—Status Enabled
  - T—Termtype
  - E—End of Record
  - T—Transmit Binary
  - E—ECHOS
  - S—Suppress Go Ahead
  - T—Timemark
- **TN3270E Connection Function Flags**, indicating what TN3270E options were negotiated ON with the client. Possible TN3270E connection function flags are:
  - B—Bind Image
  - S—SysRequest
  - R—Responses
  - S—SCS Control Codes

– D—Data Stream Control Codes (3270)

These additional header and connection values might also appear.

- **Header Abbreviations**
  – AR—Type of application definition, either **A**LLOWAPPL or **R**ESTRICTAPPL
  – LU—Indicates there are **LU**s defined on the allow or restrict appl statement
  – QS—Indicates that QSESSion was specified
  – DI—Indicates that DISCONNECTABLE was specified on the application definition
  – TP—Type of LU, either **T**erminal or **P**rinter
  – STATUS (ST)—Indicates the status of the connection
    - Negotiate in Progress—In the middle of options negotiation with the client
    - Session Pending—Negotiations are complete, but a VTAM session has not been established
    - Session Active—A VTAM session has been established
- **Connection Protocol (PR)**
  – NEGOTIAT— indicates that mode has not yet been established
  – TN3270E
  – TN3270
  – LINEMODE
  – TRANSFORM
  – DBCS
  – BINARY
- **Connection Type (TY)**
  – S-This field is blank for basic ports.
  – NS-For secure ports, see the Encryption Type abbreviations below.
- **Port Separator Lines**

  Profile, connection, and port-related displays contain a port description line that identifies the port for the subsequent data. The general format of this line is:

  ```
  ----- PORT:   nbr  status   type
  ```

  where:

  nbr is the port number

  status is either ACTIVE (accepting new connections) or QUIESCE (a VARY QUIesce has been issued for this port and new connections are not being accepted)

  type BASIC or SECURE

  Note: In some displays, the profile number (or CURR for the current profile) will also be shown on the port separator line.
- **Encryption Type Abbreviations**
  – NN—SSL_NULL_Null
  – NM—SSL_NULL_MD5
  – NS—SSL_NULL_SHA
  – 4E—SSL_RC4_MD5_EX
  – 4M—SSL_RC4_MD5
  – 4S—SSL_RC4_SHA

- 2E—SSL_RC2_MD5_EX
- DS—SSL_DES_SHA
- 3S—SSL_3DES_SHA

The Profile Group Displays

All commands containing the ″PROFILE=″ parameter are considered part of the profile group because the commands categorize (and display) the information based on what profile it is contained in.

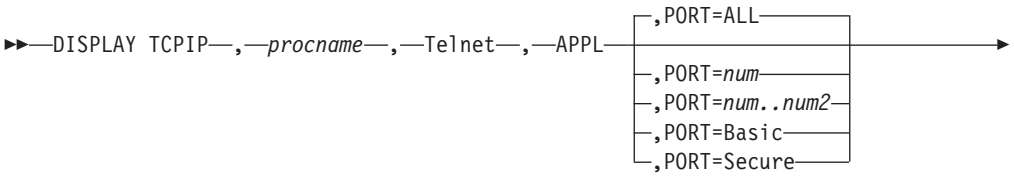
All of these commands will search all profiles that match the ″PROFILE=″ search criteria. Once a match is found the other parameters will be used to determine what is displayed for the profile.

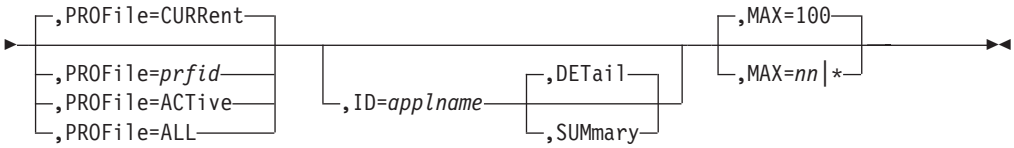
APPL Display Command

The APPL DISPLAY command is useful for identifying application problems in Telnet operation because it allows the system administrator to:

- Determine what applications are being used, how many users are actively using each one, and whether the application is restricted (RESTRICTAPPL) or allowed (ALLOWAPPL). The applications displayed are based on the ALLOWAPPL or RESTRICTAPPL statements and not actual applications with Telnet sessions.
- Determine what ALLOWAPPL and RESTRICTAPPL information is specified in a profile
- Focus on a particular application

Syntax





Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,APPL
The application keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,ID=*applname*
This DISPLAY option allows the system administrator to focus in on a particular application.

,DETail|SUMmary
DETail displays all of the defined LUs (and user IDs if it is RESTRICTAPPL) and an indication of which LUs are already being used.

SUMmary gives the same information as if ID= were not specified, but limits its scope to the specific application listed, giving a more compact display. DETail is the default.

,MAX=100|*nn*|*****

The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

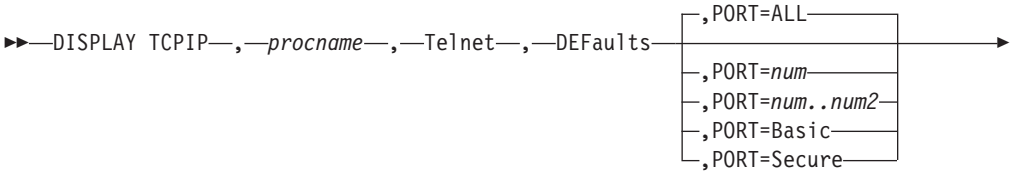
D TCPIP,,T,APPL

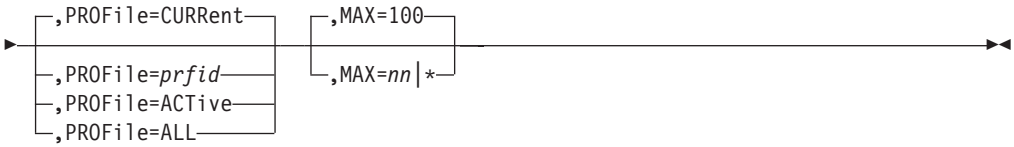
```
EZZ6062I TELNET APPL DISPLAY
         ALQD ACTIVE T DEFAULTAPPL      T LINEMODEAPPL
APPLID   RUSI CONNS  Y IP/HN GROUP      Y IP/HN GROUP
-------- ---- ------ - ---------------- - ----------------
----- PORT:   306  ACTIVE   BASIC           PROF: CURR
TSO      A---      0 I IPGRP1
TSO10    A---      0                     H TEST7.RALEIGH.IB
                                             M.COM
TSO9     A---      0                     H HNGRP2
TEST3    A---      0 H HNGRP5
ECHO01A  A-Q-      0 H IBM14-114-205-15
                       4.US.IBM-DIAL.IB
                       M.COM
ECHO01D  A-QD      0 I 9.9.9.9
TSO*     A--D      0
*        A--D      0
*NOAPPL* ----      3
TOTAL USERS        3
14 OF 14 RECORDS DISPLAYED
```

DEFAULTS Display Command

The DEFAULTS DISPLAY command allows the system administrator to verify that the profile information for the DEFAULTLUS, LINEMODEAPPL, USSTCP, and INTERPTCP statements is correct and to understand what was specified in previous profiles when the actual file may no longer be available.

Syntax





Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,DEFaults
The defaults keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,MAX=100|*nn***|***
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

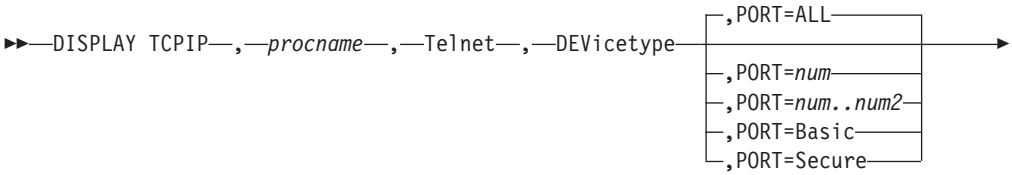
```
D TCPIP,,T,DEFAULTS

EZZ6070I TELNET DEFAULTS DISPLAY
APPLID/
TABLE     TYPE    REFTYPE  REFERENCE
--------  ------  -------  ------------------------------
----- PORT:   306  ACTIVE   BASIC           PROF: CURR
TEST3     DEFLT   HNGRP    HNGRP5
TSO       DEFLT   IPGRP    IPGRP1
TSO5      DEFLT   HSTNAME  TEST1.RALEIGH.IBM.COM
ECHO01A   DEFLT   HSTNAME  IBM14-114-205-154.US.IBM-DIAL.I
                           BM.COM
```

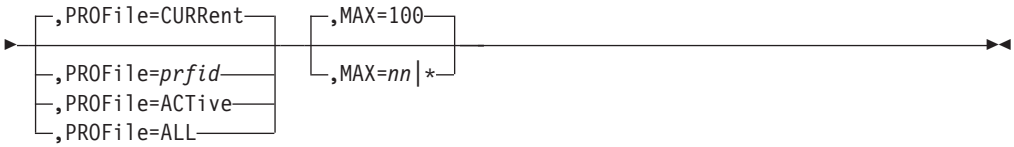
```
ECHO01D   DEFLT   IPADDR   255.255.255.255:9.9.9.9
TSO11     DEFLT   IPADDR   0.0.0.0:0.0.0.0
TSO6      DEFLT   HSTNAME  TEST6.RALEIGH.IBM.COM
TSO11     LINEMD  IPADDR   0.0.0.0:0.0.0.0
TSO9      LINEMD  HNGRP    HNGRP2
TSO10     LINEMD  HSTNAME  TEST7.RALEIGH.IBM.COM
EZBTPUST  USSTCP  HNGRP    HNGRP1
                  HNGRP    HNGRP2
                  IPGRP    IPGRP1
                  HSTNAME  TEST7.RALEIGH.IBM.COM
16 OF 16 RECORDS DISPLAYED
```

DEVICETYPE Display Command

The DEVICETYPE DISPLAY command allows the system administrator to verify that the profile information for any DEVICETYPE statement is correct and understand what was specified in previous profiles when the actual file may no longer be available. Since all Telnet device types have a default which is used if no DEVICETYPE statement in the PROFILE/OBEYFILE specifically changes it, the display indicates whether a default was replaced by a TELNETDEVICE statement.

Syntax





Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,DEVicetype
The device type keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,MAX=100|*nn***|***
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

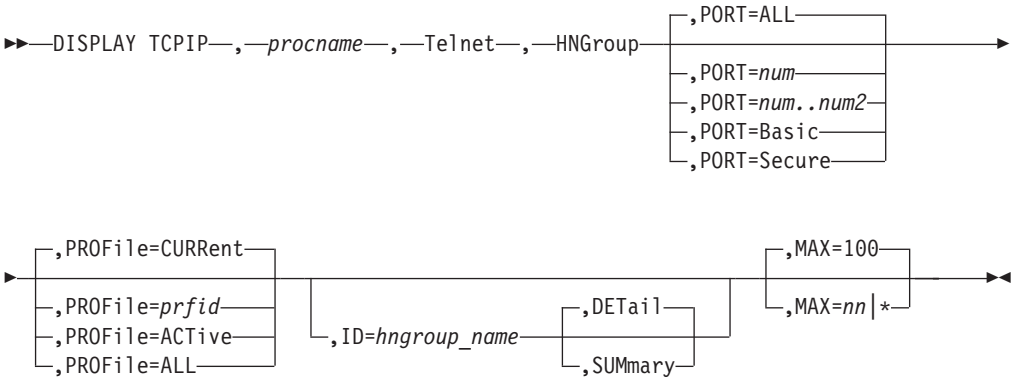
```
D TCPIP,TCPCS4,T,DEVICE,MAX=10

EZZ6068I TELNET DEVICETYPE DISPLAY 456
TELNET        TN3270    TN3270E   DEFAULT
DEVICE TYPE   LOGMODE   LOGMODE   REPLACED
------------  --------  --------  ---------
----- PORT:   911  ACTIVE   SECURE          PROF: CURR
IBM-3277      D4B32782  *N/A*       N   -
IBM-3278-2-E  NSX32702  SNX32702    N   N
IBM-3278-2    D4B32782  SNX32702    N   N
```

```
IBM-3278-3-E  NSX32702  SNX32703    N   N
IBM-3278-3    D4B32783  SNX32703    N   N
IBM-3278-4-E  NSX32702  SNX32704    N   N
IBM-3278-4    D4B32784  SNX32704    N   N
IBM-3278-5-E  NSX32702  SNX32705    N   N
IBM-3278-5    D4B32785  SNX32705    N   N
10 OF 84 RECORDS DISPLAYED
```

HNGROUP Display Command

The HNGROUP DISPLAY command allows the system administrator to determine what HN groups are being used and how many users are actively using each one. The number of active users on each HN group helps in determining if more HNs than expected are being mapped to a particular HN group. This information, combined with the WHEREUSED information for this HN group, can determine if more LUs should be defined to better service an HN group.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,HNGroup
The HN Group keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,ID=*hngroup_name*
This DISPLAY option allows the system administrator to focus in on a particular HN group.

DETail|SUMmary
The DETail parameter displays what HNs accessed the system through this HN group and what masked HNs account for any unexpected connections to this HN group. This is done by displaying all of the defined HNs/wildcard HNs and indicating which HNs are being actively used.

SUMmary gives the same information as if ID= were not specified, but limits its scope to the specific HN group listed, giving a more compact display. DETail is the default.

,MAX=100|*nn*|*****

The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

D TCPIP,TCPCS6,T,HNG

```
EZZ6076I TELNET HNGROUP DISPLAY
HNGROUP           ACTIVE
NAME              HNS
----------------  ------
----- PORT:   306  ACTIVE   BASIC           PROF: CURR
HNGRP1                 3
HNGRP2                 0
HNGRP3                 0
HNGRP5                 0
5 OF 5 RECORDS DISPLAYED
```

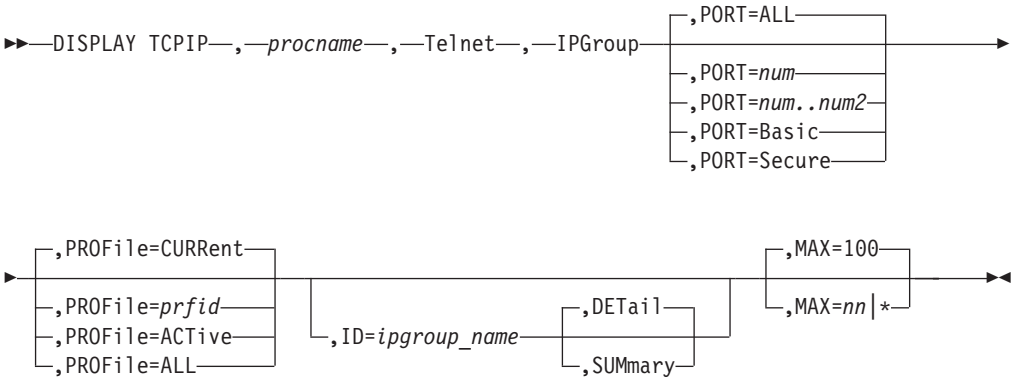
D TCPIP,TCPCS6,T,HNG,ID=HNGRP1

```
EZZ6077I TELNET HNGROUP DISPLAY
DETAIL GROUPNAME = HNGRP1
----- PORT:   306  ACTIVE   BASIC           PROF: CURR
HNGROUP        ACTIVE HNS =     3
  EXACT DEFINED HOSTNAMES
      TEST1.RALEIGH.IBM.COM
      TEST2.RALEIGH.IBM.COM
      TEST3.IBM.COM
      TEST4.RALEIGH.IBM.COM
     +TROTTMAN.RALEIGH.IBM.COM
8 OF 8 RECORDS DISPLAYED
```

IPGROUP Display Command

The IPGROUP DISPLAY command allows the system administrator to determine what IP groups are being used and how many users are actively using each one. The number of active users on each IP group helps in determining if more IPs than expected are being mapped to a particular IP group. This information, combined with the WHEREUSED information for this IP group, can determine if more LUs should be defined to better service an IP group.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,IPGroup
The IP Group keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,ID=*ipgroup_name*
This DISPLAY option allows the system administrator to focus in on a particular IP group.

DETail|SUMmary
The DETail parameter displays what IPs accessed the system through this IP group and what masked IPs account for any unexpected connections to this IP group. This is done by displaying all of the defined IPs/masked IPs and indicating which IPs are being actively used.

SUMmary gives the same information as if ID= were not specified, but limits its scope to the specific IP group listed, giving a more compact display. DETail is the default.

,MAX=100|*nn*|*

The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

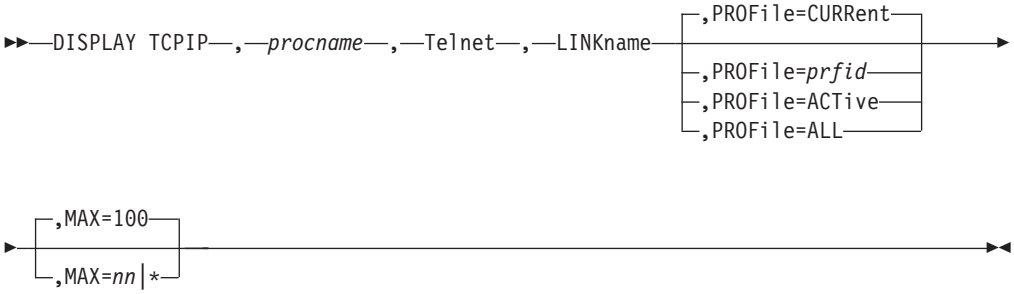
D TCPIP,TCPCS4,T,IPG,PROF=ALL

```
EZZ6074I TELNET IPGROUP DISPLAY 334
IPGROUP           ACTIVE
NAME              IPS
----------------  ------
----- PORT:   921  ACTIVE   BASIC           PROF: CURR
IPGRP67                0
IPGRP80                0
BOGUS                  0
BWS                    0
----- PORT:   931  ACTIVE   BASIC           PROF: CURR
IPGRP67                0
IPGRP80                0
BOGUS                  0
BWS                    0
----- PORT:   931  ACTIVE   BASIC           PROF: 0001
IPGRP67                2
IPGRP80                0
BOGUS                  0
BWS                    0
15 OF 15 RECORDS DISPLAYED
```

LINKNAME Display Command

The LINKNAME DISPLAY command allows the system administrator to determine what linknames are being used and how many users are actively using each one. The number of active users on each linkname helps in determining if more IPs than expected are being mapped to a particular linkname.

Syntax



Parameters

,procname
: The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
: Directs the command to the Telnet component.

,LINKname
: The link name keyword.

,PROFile =CURRent|*prfid*|ACTIVE|ALL
: *CURRent* is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,MAX=100|*nn*|*
: The number of data lines displayed. The default is 100. The ″*″ means ″all″.

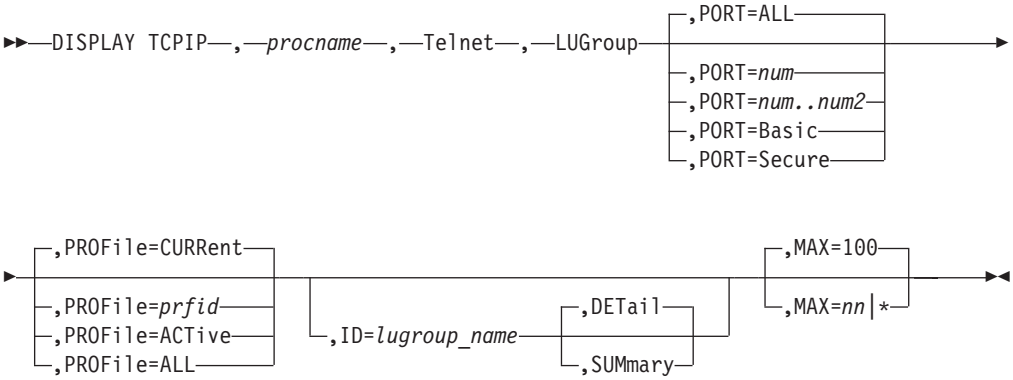
Examples

```
D TCPIP,TCPCS,T,LINK,PORT=ALL,PROF=ALL

EZZ6071I TELNET LINKNAME DISPLAY 817
                            ACTIVE
LINKNAME  IPADDR            USERS
--------  ---------------   ------
----- PORT:    33  ACTIVE   BASIC           PROF: CURR
CTCLNK3   9.67.116.144         0
----- PORT:    63  ACTIVE   BASIC           PROF: CURR
CTCLNK3   9.67.116.144         0
----- PORT:   992  ACTIVE   SECURE          PROF: CURR
CTCLNK3   9.67.116.144         0
----- PORT:  1123  ACTIVE   SECURE          PROF: CURR
CTCLNK3   9.67.116.144         0
8 OF 8 RECORDS DISPLAYED
```

LUGROUP Display Command

The LUGROUP DISPLAY command allows the system administrator to determine what LUGROUPs and PRTGROUPs are being used, the number of LUs that are defined for each one, and how many of those LUs are already in use.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,LUGroup
The LU group keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,ID=*lugroup_name*
This DISPLAY option allows the system administrator to focus in on a particular LUGROUP or PRTGROUP.

,DETail|SUMmary
The DETail option displays the LUs defined for the LUGROUP or PRTGROUP and indicates which LUs are active. NOTE: Some LUs may indicate not active in the current profile because they are active in an earlier profile.

SUMmary gives the same information as if ID= were not specified, but limits its scope to the specific LUGROUP or PRTGROUP listed, giving a more compact display. DETail is the default.

,MAX=100|*nn|**
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

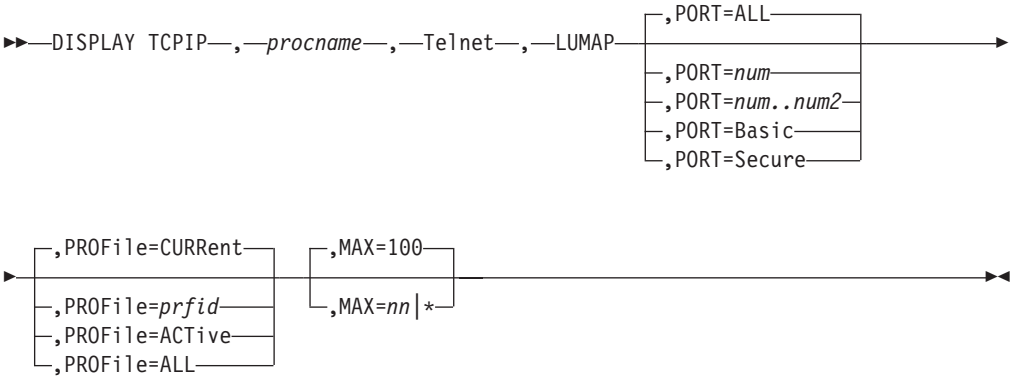
D TCPIP,TCPCS4,T,LUG,PROF=ALL

```
EZZ6072I TELNET LUGROUP DISPLAY
LUGROUP         DEFINED  ACTIVE
NAME      TYPE  LUS      LUS
--------  ----  ------   ------
----- PORT:   911  ACTIVE   SECURE          PROF: CURR
*DEFLUS*  LU        99        0
----- PORT:   921  ACTIVE   BASIC           PROF: CURR
LUGRP1    LU        51        0
*DEFLUS*  LU       100        0
PRTGRP1   PRT        1        0
----- PORT:   931  ACTIVE   BASIC           PROF: CURR
LUGRP1    LU        51        0
*DEFLUS*  LU       100        0
PRTGRP1   PRT        1        0
----- PORT:  1099  ACTIVE   BASIC           PROF: CURR
*DEFLUS*  LU         5        2
----- PORT:  1099  ACTIVE   BASIC           PROF: 0001
*DEFLUS*  LU         5        1
18 OF 18 RECORDS DISPLAYED
```

LUMAP Display Command

The LUMAP DISPLAY command allows the system administrator to determine what LUMAPs and PRTMAPs are defined.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,LUMap
The LU map keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid*|ACTIVE|ALL
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,MAX=100|*nn*|*
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

D TCPIP,,T,LUMAP

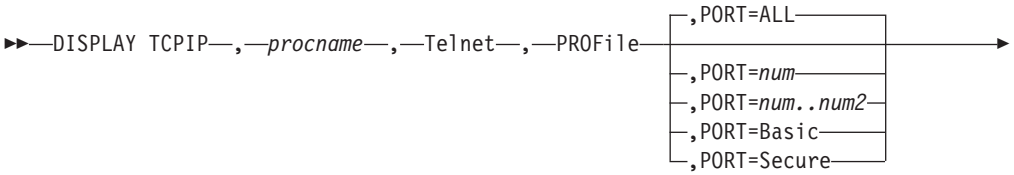
```
EZZ6069I TELNET LUMAP DISPLAY
LUGROUP         T                           PRTGROUP
NAME      TYPE  Y  IP/HN GROUP       SPEC  NAME
--------  ----  -  ----------------  ----  --------
----- PORT:   306  ACTIVE   BASIC           PROF: CURR
LUGRP1    LU    H  HNGRP1            YES
PRTGRP1   PRT   I  IPGRP1            NO
----- PORT:   336  ACTIVE   SECURE          PROF: CURR
LUGRP2    LU    H  HNGRP1            NO    PRTGRP2
5 OF 5 RECORDS DISPLAYED
```

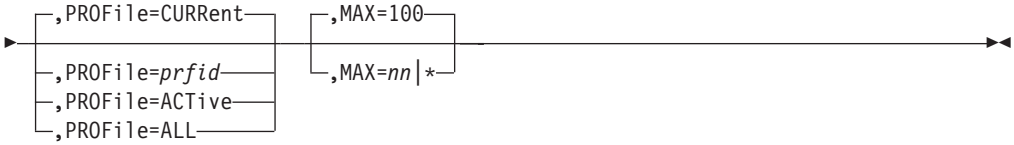
PROFILE Display Command

The PROFILE DISPLAY command allows the system administrator to:

- Determine what profile-wide options are in effect for each profile
- See which profiles are still being used
- Find out how many users are on each profile

Syntax





Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,PROFile
The profile keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
CURRent is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,MAX=100|*nn***|****
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

```
D TCPIP,,T,PROF

EZZ6060I TELNET PROFILE DISPLAY 796
 PROF S TOTAL  PROFILE       SMF   ---------- TIMEOUTS ---------
 ID   T USERS  OPTIONS     INT TRM INACT PRINT SSLTO TMARK SCANI
 ---- - ------ ----------- --- --- ----- ----- ----- ----- -----
 ----- PORT:   340  ACTIVE   SECURE
 CURR A      3 --------W-    0   0   600     0    10 20000 10000
               NN,NM,NS,4E,4M,--,2E,DS,3S            :SAFCERT
 ----- PORT:   930  ACTIVE   BASIC
 CURR A      0 -----T----    0   0     0     0     5 20000 10000
```

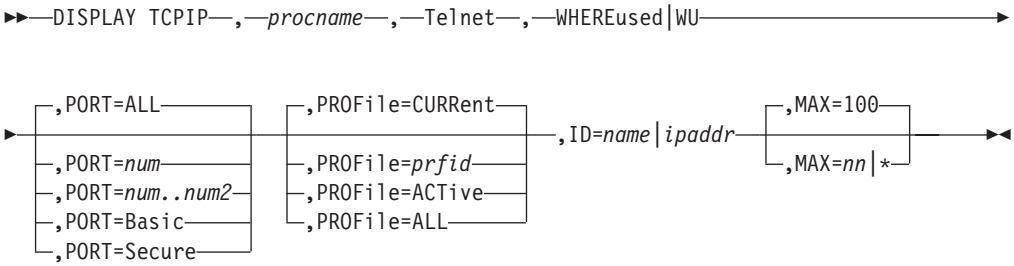
```
----- PORT:   931  ACTIVE   BASIC
CURR A      0 --------W-    0   0     0     0     5 20000 10000
TOTAL       3
KEYRING   MVS 'USER40.PAJ0428.KYR'
8 OF 8 RECORDS DISPLAYED
```

WHEREUSED Display Command

The WHEREUSED DISPLAY command allows the system administrator to determine all the places where a particular name or IPaddr was used in the profile. It is a very powerful command used to determine:

- How a particular IP address could get access to the host through TELNET
- Whether an luname is assigned to too many groups or applications
- Whether a name is being used in too many places so that its use is confusing
- Whether the profile is correctly using the same name in several places
- Whether a name you plan on adding is unique

Syntax



Parameters

,procname
: The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
: Directs the command to the Telnet component.

,WHEREused|WU
: The whereused keyword.

PORT
: Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,PROFile =CURRent|*prfid***|ACTIVE|ALL**
: *CURRent* is the name of the current profile. *prfid* is the profile ID. *ACTIVE* is all the active profiles. *ALL* is all profiles, both active and inactive. CURRent is the default.

,ID=*name***|***ipaddr*
: The name or IP address to use in the search. The IP address should be specified in the dotted decimal format.

,MAX=100|*nn***|***
: The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

```
D TCPIP,TCPCS4,T,WU,PORT=S,ID=9.67.116.29

EZZ6066I TELNET WHEREUSED DISPLAY 487
IPADDR  = 9.67.116.29
WHEREUSED         NAME
----------------  -----------------
```

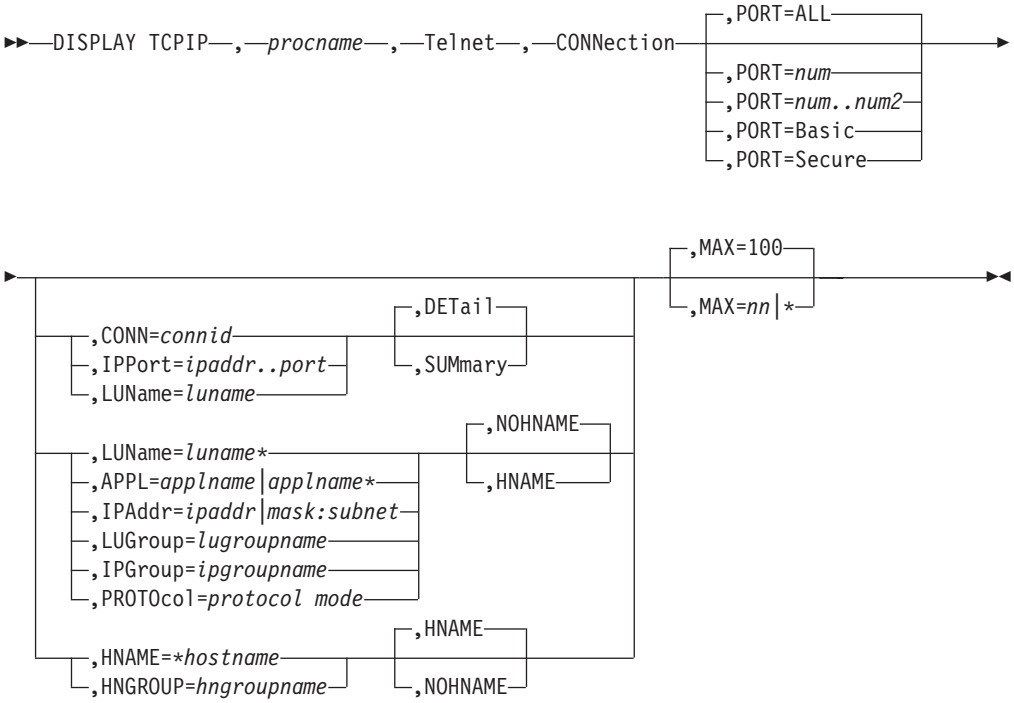
```
----- PORT:   911  ACTIVE   SECURE          PROF: CURR
DEFAULTAPPL       APPL2
2 OF 2 RECORDS DISPLAYED
```

The Connection Group Displays – CONNECTION Display Command

The CONNECTION Display command with the SUMmary parameter specified allows the system administrator to take a high-level view of what connections exist and what they are being used for.

The DISPLAY command with the DETail parameter specified gives the system administrator a complete look at one connection. It will get all of the information available regarding a single connection.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,CONNection
The connection keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,CONN=*connid*
Displays detailed information about a specific TCPIP connection ID.

,IPPort=*ipaddr..port*
Displays detailed information about a specific IP port and address.

,LUName=*luname**

The name of the LU for which you are searching. The wildcard (*) is allowed only as the last character of the LUName. If no (*), a detailed display will appear.

,SUMmary|DETail

DETail displays all of the information about the requested connection.

SUMmary displays a subset of the information about the requested connection.

,APPL=*applname|applname**

The application name of the application for which you are searching. The wildcard (*) is allowed only as the last character.

,IPAddr=*ipaddr|mask:subnet*

The IP address of the connection for which you are searching. The `mask:subnet` designation is essentially allowing an IP wildcard.

,LUGroup=*lugroupname*

The name of the LU group for which you are searching.

,IPGroup=*ipgroupname*

The name of the IP group for which you are searching.

,PROTOcol=*protocol mode*

The protocol mode for which you are searching.

,HNAME|NOHNAME

The summary display includes client host names when HNAME is specified. The summary display omits client host names when NOHNAME is specified.

,HNAME=*hostname*

The host name for which you are searching. Single or double asterisks are permitted as wildcards:

- Use a single asterisk (*) to indicate that any value is acceptable for a particular qualifier in a particular position within the host name. For example, *.*.IBM.COM matches USER1.RALEIGH.IBM.COM, but does not match USER1.TCP.RALEIGH.IBM.COM since this name includes an extra qualifier.
- Use a double asterisk (**) to indicate that any number of qualifiers are acceptable to the left of the asterisks. For example, **.IBM.COM matches USER1.IBM.COM, USER1.RALEIGH.IBM.COM, and USER1.TCP.RALEIGH.IBM.COM.

Both wildcard techniques require that the entire qualifier be wildcarded. For example, *USER.IBM.COM is not a valid use of a wildcard. In this case, use *.IBM.COM instead.

,HNGROUP=*hngroupname*

The name of the HN group for which you are searching.

,MAX=100|*nn|**

The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Usage Notes

Only one connection at a time will be displayed with parameters CONN=, IPPort=, and LUName= if no wildcard is on LUName.

Examples

```
D TCPIP,TCPCS4,T,CONN

EZZ6064I TELNET CONNECTION DISPLAY 492
      EN                                          TSP
CONN  TY IPADDR..PORT           LUNAME   APPLID   PTR LOGMODE  OPTIONS
----- -- --------------------- -------- -------- --- -------- -------
      PORT:   931  ACTIVE   SECURE          PROF: 0001
0003C DS  9.37.82.196..1089      TCPM1100 APPL4    TAT D4B32782 ----ES-
0003B 3S  9.37.82.196..1088      TCPM1101 APPL4    TAT D4B32782 ----ES-
      PORT:  1099  ACTIVE   BASIC           PROF: CURR
00033     9.37.82.9..1420        TCPM1012 TSO10005 TAE SNX32702 ETET---
      PORT:  1099  ACTIVE   BASIC           PROF: 0001
0002F     9.37.82.9..1419        TCPM1011 TSO10004 TAE SNX32702 ETET---
7 OF 7 RECORDS DISPLAYED
```

```
D TCPIP,TCPCS4,T,CONN,CONN=3C
EZZ6065I TELNET CONNECTION DISPLAY 500
  CONN: 0003C    IPADDR..PORT: 9.37.82.196..1089
  CONNECTED: 17:00:40  03/02/98  STATUS: SESSION ACTIVE
  PORT:   931  ACTIVE    SECURE       ACCESS: SECURE DS clientauth
  PROFILE ID: 0001   PROFILE OPTIONS: --L-MT---
  PROTOCOL: TRANFORM LOGMODE: D4B32782 DEVICETYPE: IBM-3278-2
    OPTIONS: ----ES-  3270E FUNCTIONS: *N/A*
  USSTABLE: **N/A**     IPGROUP: **N/A**
  LUNAME: TCPM1100      TYPE: TERMINAL
    LU/PRTGROUP: *DEFLUS*    IPGROUP: **N/A**
  APPLID: APPL4
    DEFAULTAPPL IPGROUP: 0.0.0.0
    RESTRICTAPPL USERID: **N/A**        Certuserid
12 OF 12 RECORDS DISPLAYED
```

In the example directly above, *clientauth* indicates one of the following types of client authorization completed for this connection:

- Null (no client authentication requested)
- SSLCERT
- SAFCERT (certificate defined to the security product, but port-specific authorization not requested)
- SAFAUTH (authorized in a SERVAUTH class profile that covered this port)

Also in this example, *certuserid* is displayed if CLIENTAUTH SAFCERT was specified for the port.

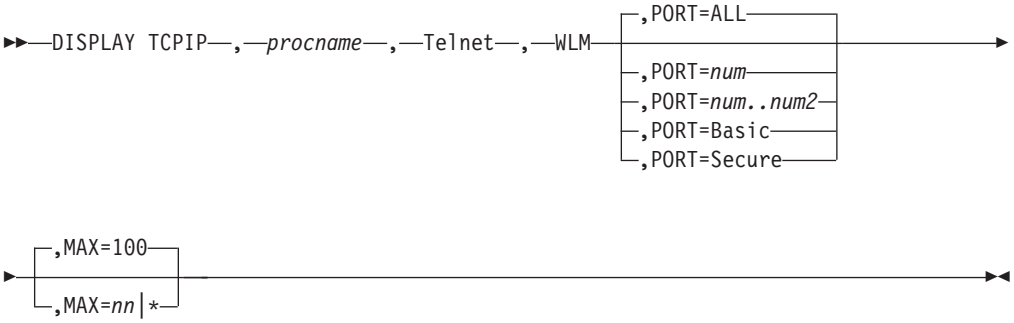
- CERT.USERID: *userid*, where *userid* is the userid associated with the client certificate registered with the SAF compliant security product.
- Null

The Port Group Displays – WLM Display Command

The WLM DISPLAY command allows the system administrator to determine what names Telnet has used to register itself with the Workload Manager (WLM) and which of these registered names is available to users. These names are only available to clients when DNS is configured for Connection Optimization. See the OS/390 SecureWay Communications Server: IP Configuration for more information about Connection Optimization.

Since TELNET in a QUIESCE state will unregister itself, the system administrator can also use this command to determine whether TELNET is in a QUIESCE or STARTed state.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component.

,WLM
The workload manager keyword.

PORT
Specifies that a specific port number, port number range, all active ports, all secure ports, or all basic ports are to be displayed.

,MAX=100|*nn*|*****
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

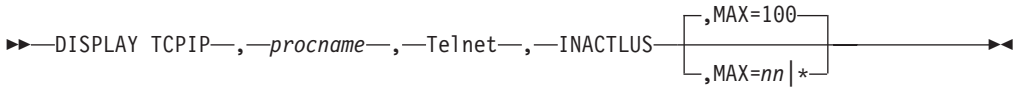
Examples

```
D TCPIP,TCPCS4,T,WLM

EZZ6067I TELNET WLM DISPLAY 638
WLM CLUSTER NAME          STATUS
------------------  ---------------------
----- PORT:  2026  ACTIVE   BASIC

WLM2                REGISTERED
WLM1                REGISTERED
3 OF 3 RECORDS DISPLAYED
```

The Server Group Displays - INACTLUS Display Command

The INACTLUS Display command allows a system administrator to see all of the LUs that are not available to any users because the VARY INACT command was issued or the OPEN ACB failed and Telnet automatically sets the LU inactive.

Syntax



Parameters

,procname
The member name of the cataloged procedure used to start the TCPIP address space.

,Telnet
Directs the command to the Telnet component

,INACTLUS
The inactive LUs keyword

,MAX=100|*nn*|*****
The number of data lines displayed. The default is 100. The ″*″ means ″all″.

Examples

```
D TCPIP,,T,INACTLUS

EZZ6061I TELNET INACTLUS DISPLAY
INACTIVE LUS
               TCPM1003  TCPM1005  TCPM1004  TCPM1001  TCPM1010
               TCPM1015  TCPM1012  TCPM1008
2 OF 2 RECORDS DISPLAYED
```

Chapter 14. Configuring the File Transfer Protocol (FTP) Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the File Transfer Protocol (FTP) server. It also provides information about using the MODIFY command to dynamically control tracing. Information about security considerations for the FTP server is described in “Security Considerations for the FTP Server” on page 545.

Configuration Process

Steps to configure the FTP server:

1. Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS information.
2. Update ETC.SERVICES with the port to be used by the FTP server.
3. Update the FTPD cataloged procedure *hlq*.SEZAINST(FTPD). FTPD is also known by the SMP/E distribution name EZAFTPAP.
4. Specify FTP configuration statements in FTP.DATA.
5. Configure the FTP server for SMF (optional).
6. Configure the FTP user-written exits (optional).
7. Specify configuration statements in TCPIP.DATA.
8. Install the SQL Query Function (optional) and access the DB2 modules.
9. Update /etc/syslog.conf for the FTP server.

To dynamically enable or disable the trace options, see “Starting, Stopping, and Tracing the FTP Server” on page 610.

Step 1: Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS Information

If you want the FTP server to be started automatically when the TCPIP address space is started, then include the name of the member containing the FTP server catalogued procedure in the AUTOLOG statement. For example, if your procedure is called FTPD:

```
AUTOLOG
  FTPD JOBNAME FTPD1
ENDAUTOLOG
```

To reserve ports 21 and 20 for the FTP server, add the following:

```
PORT
   21 TCP FTPD1             ; FTP server control port
   20 TCP OMVS NOAUTOLOG    ; FTP server data port
```

To allow the FTP data connections to timeout when there has been no activity on the data connection for a certain amount of time, add the KEEPALIVEOPTIONS statement to the PROFILE.TCPIP data set:

```
KEEPALIVEOPTIONS INTERVAL number_of_minutes ENDKEEPALIVEOPTIONS
```

Be careful when choosing a timeout interval for the KEEPALIVEOPTIONS statement because this value will affect *all TCP* connections at this host for which KEEPALIVEOPTIONS has been activated, not just the FTP data connections.

See “AUTOLOG Statement” on page 137 for more information on the AUTOLOG statement. See “PORT Statement” on page 229 for more information on the PORT statement. See “KEEPALIVEOPTIONS Statement” on page 222 for more information on the KEEPALIVEOPTIONS statement.

Step 2: Update ETC.SERVICES

The ETC.SERVICES data set contains the relationship between service names (servers) and port numbers in the OS/390 UNIX environment. Update your ETC.SERVICES file to specify the control port that the FTP server is to use. For example, add the following

```
ftp 21/tcp
```

A search order used to find the ETC.SERVICES file. See “Chapter 1. Before You Begin” on page 7 for a description of this search order.

Update your ETC.SERVICES file to specify the control point that the FTP server is to use. For example, add the following:

```
ftp 21/tcp
```

Notes:

1. In the ETC.SERVICES file, only one port (the one for the control connection) is listed.
2. The port specified for FTP in the ETC.SERVICES file can be overridden by the FTP start parameter PORT nnnn.

Step 3: Update the FTPD Cataloged Procedure

Update the FTP cataloged procedure FTPD by copying the sample in *hlq*.SEZAINST(FTPD) to your system or recognized PROCLIB and modifying it to suit your local configuration. Specify FTPD parameters and change the data set names as required.

The FTP cataloged procedure is also known by the SMP/E distribution name EZAFTPAP.

FTP Server Cataloged Procedure (FTPD)

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//*********************************************************************
//*         Descriptive Name:            FTP Server Start Procedure   *
//*         File Name:                   tcpip.SEZAINST(EZAFTPAP)     *
//*                                      tcpip.SEZAINST(FTPD)         *
//*         SMP/E Distribution Name:     EZAFTPAP                     *
//*                                                                   *
//*         Licensed Materials - Property of IBM                      *
//*         "Restricted Materials of IBM"                             *
//*         5647-A01                                                  *
//*         (C) Copyright IBM Corp. 1995, 1999                        *
//*         Status = CSV2R8                                           *
//*********************************************************************
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON)/&PARMS'
//*   PARM=('POSIX(ON) ALL31(ON)',
```

```
//*  'ENVAR("RESOLVER_CONFIG=//''TCPIVP.TCPPARMS(TCPDATA)''")/&PARMS')
//*
//**** IVP Note ********************************************************
//*
//* If executing the FTP installation verification procedures (IVP),
//* - Comment the first PARM card and uncomment both lines of the
//*   second PARM card
//* - Uncomment the appropriate SYSFTPD and SYSTCPD DD cards for the IVP
//*
//*********************************************************************
//*       The C runtime libraries should be in the system's link
//*       list or add them to the STEPLIB definition here.  If you
//*       add them to STEPLIB, they must be APF authorized.
//*
//*       To submit SQL queries to DB2 through FTP, the DB2 load
//*       library with the suffix DSNLOAD should be in the system's
//*       link list, or added to the STEPLIB definition here.  If
//*       you add it to STEPLIB, it must be APF authorized.
//*
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*
//*           SYSFTPD DD statement
//*           jobname.FTP.DATA
//*           /etc/ftp.data
//*           SYS1.TCPPARMS(FTPDATA)
//*           tcpip.FTP.DATA
//*
//*       If no FTP.DATA file is found, FTP default values are used.
//*       For information on FTP defaults, see OS/390 eNetwork
//*       Communications Server: IP Configuration.
//*SYSFTPD DD DISP=SHR,DSN=TCPIP.SEZAINST(FTPSDATA)
//*SYSFTPD DD DISP=SHR,DSN=TCPIVP.TCPPARMS(FTPSDATA)
//*
//*       SYSTCPD explicitly identifies which file is to be
//*       used to obtain the parameters defined by TCPIP.DATA.
//*       The SYSTCPD DD statement should be placed in the JCL of
//*       the server.  The file can be any sequential data set,
//*       member of a partitioned data set (PDS), or HFS file.
//*SYSTCPD DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
//*SYSTCPD DD DISP=SHR,DSN=TCPIVP.TCPPARMS(TCPDATA)
//*
```

The REGION size requirement for the FTPD address space might increase under certain circumstances.

Specifying the FTPD Parameters

The system parameters required by the FTP server are passed by the PARM parameter on the EXEC statement of the FTPD cataloged procedure. Add your parameters to `PARMS='` in the PROC statement of the FTPD cataloged procedure, making certain that:

- Each parameter is separated by a blank
- All parameters are in uppercase

For example: `//FTPD PROC MODULE='FTPD',PARMS='TRACE ANONYMOUS PORT 621'`

ANONYMOUS
Allows remote users to enter ANONYMOUS as a user ID and log on without

supplying a logon password. Specifying ANONYMOUS makes your universally permitted data sets accessible to all users on the TCP/IP network.

ANONYMOUS=*user_id*
Allows a remote user to enter ANONYMOUS as a user ID. When ANONYMOUS is entered as the user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The user will be prompted for the password to *user_id* and, if the user enters the correct password, the user will be logged in as the specified *user_ID*.

ANONYMOUS=*user_id/password*
Allows a remote user to enter ANONYMOUS as a user ID. When ANONYMOUS is entered as the user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The FTP server automatically provides the *password* for the specified *user_id* and the user will be logged in as the specified *user_ID*.

AUTOMOUNT
Permits a DASD volume to be mounted when attempts are made to access data sets on that volume.

AUTORECALL
Permits data sets migrated by a storage manager, such as Hierarchical Storage Manager (HSM), to be recalled automatically.

DATASETMODE
Treats all lower qualifiers of address space names as part of the same directory. This affects the behavior of DIR, LS, MGET, and MDLETE because all lower qualifiers are returned.

DIRECTORYMODE
Treats each level of an address space name as if it were a directory. This affects the behavior of DIR, LS, MGET, and MDLETE because only the next lower qualifier is returned.

INACTIVE *number_seconds*
Sets the inactivity time-out to the specified number of seconds. A control connection that is inactive for this amount of time is closed. The default inactivity time-out is 300 seconds (5 minutes). The maximum inactive time is 86 400 seconds. A value of 0 will disable the inactivity timer and inactive control connections will not time out.

NOAUTOMOUNT
Prevents a DASD volume from being mounted when attempts are made to access data sets on that volume.

NOAUTORECALL
Prevents data sets migrated by a storage manager, such as HSM, from being recalled automatically. Migrated data sets can still be deleted even though NOAUTORECALL is specified.

Note: Only sequential and whole partitioned data sets can be deleted without recalling. Partitioned data set members require the whole data set to be recalled.

PORT *port_num*
Accepts incoming requests on the specified (decimal) port number rather than the port specified in /etc/services or the default port of 21. (*port_num* – 1) will be used for data transfer. The maximum port number is 65534.

TRACE
Displays tracing information to syslog. Running TRACE will affect performance. TRACE should be specified only if you need information for the IBM TCP/IP support group.

Defining Optional Environment Variables for the FTP Server

The FTP server optionally uses environment variables to identify the translate table data sets to be used for the control and data connections. These environment variables are used to override a default naming convention as described below.

In your FTP.DATA file, you can use the CCXLATE or XLATE statements to specify a name that corresponds to a particular data set that is to be used for the initial translate tables for the control or data connections.

FTP will look for an environment variable defined as *_FTPXLATE_name=fully_qualified_dsn*, where *name* must be 1 to 8 uppercase characters or numbers, and *fully_qualified_dsn* can be a fully-qualified MVS data set name or hfs filename.

If the environment variable exists, FTP will use the data set name defined by the environment variable. If no such environment variable is defined, FTP will use the data set name *hlq.name*.TCPXLBIN.

Similarly, from any client you can issue *site xlate=name* to set the translate tables for the data connection for that particular FTP session. The FTP server will look for an environment variable called *_FTPXLATE_name*. If the environment variable does not exist, the server will look for a data set called *hlq.name*.TCPXLBIN.

Note: The CCXLATE and XLATE statements and SITE XLATE command are not case sensitive, but the name of the optional environment variable is case sensitive and must be in uppercase or FTP will not recognize it.

You can use the ENVAR runtime option in your FTPD start procedure to set environment variables for the FTP server. For information on using the ENVAR runtime option to set environment variables, see *OS/390 C/C++ Programming Guide*. The following example shows how to specify environment variables in your FTPD started procedure:

```
//      PARM=('POSIX(ON)  ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIP"',
//           ' "TZ=EST")/&PARMS')
```

If you need to set several environment variables, you can use the BPXBATCH program to start FTPD. This program is documented in *OS/390 V2R5.0 Command Reference*.

Example: The following shows a JCL example:

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE"',
//           '"TZ=EST")/&PARMS')
```

Step 4: Specify FTP Configuration Statements in FTP.DATA

The FTP.DATA data set is optional. The FTP daemon looks for this data set during initialization, following this sequence:

1. A data set specified by the //SYSFTPD DD statement
2. *ftpserve_job_name*.FTP.DATA
3. /etc/ftp.data
4. SYS1.TCPPARMS(FTPDATA)
5. *hlq*.FTP.DATA data set

If you use an MVS data set, this data set should have a logical record length of 80 and a block size that is a multiple of 80.

The default values for the FTP server parameters are in the FTPD module. You can change these defaults using statements in the FTP.DATA configuration data set.

It is not necessary to include all statements in the FTP.DATA data set. Only include the statements if the default value is not what you want, since the default will be used for any statement not included in the FTP.DATA data set.

Several of the FTP server parameters can be changed during an FTP session by issuing the SITE subcommand from the FTP client. See *OS/390 SecureWay Communications Server: IP User's Guide* for more information.

The FTP.DATA data set can also be used to change the defaults for the FTP client local site parameters. See the *OS/390 SecureWay Communications Server: IP User's Guide* for more information about using the FTP.DATA data set for the FTP client local site parameters.

Data set attributes play a significant role in FTP performance. If your environment permits, tune both BLOCKSIZE and LRECL according to the following recommendations:

- Use half a track as the block size
- For IBM 3380 DASD, use 23424 as the block size with an LRECL of 64 bytes
- For IBM 3390 or IBM 9334, use 27968 as the block size with an LRECL of 64 bytes
- Use FB as the data set allocation format
- Use cached DASD controllers
- If your environment permits, use a pre-allocated data set for FTP transfer operations into MVS

Summary of FTP Server Configuration Statements

The statements for the FTP.DATA data set are summarized in Table 21 and explained in detail in "FTP.DATA Data Set Statements" on page 546.

If you plan to share the FTP server FTP.DATA data set with the FTP client, note that some of the values for the statements in the FTP.DATA data set have different meanings in the two environments. If the files are shared, error messages might be generated or values that are not valid might be used for each client that uses the FTP.DATA data set containing server-only keywords. To avoid these errors, use separate FTP.DATA data sets for the FTP client and the FTP server if you are specifying any conflicting keywords.

Table 21. Summary of FTP Server Configuration Statements

Statement	Description	Page
ANONYMOUS	Allow a remote user to issue USER ANONYMOUS without supplying a logon password	547

Table 21. Summary of FTP Server Configuration Statements (continued)

Statement	Description	Page
ASATRANS	Specify how print control characters should be handled	549
AUTOMOUNT	Specify whether to mount DASD volumes containing data sets to be accessed	550
AUTORECALL	Automatically recall data sets migrated by the storage manager	551
AUTOTAPEMOUNT	Specify whether to mount tape volumes containing data sets to be accessed	552
BLKSIZE	Specify the block size of newly allocated data sets	553
BUFNO	Specify the number of access method buffers	554
CCXLATE	Specify the translation table data set for the control connection.	555
CHKPTINT	Specify the checkpoint interval when the FTP server is the sending site in a file transfer request	556
CONDDISP	Keep and catalog or delete a data set when a file transfer ends prematurely	557
CTRLCONN	Specify ASCII codeset to be used for the control connection	558
DATACLASS	Specify the SMS-managed data class as defined by your organization for the FTP server	559
DB2	Specify the name of the DB2 subsystem	561
DB2PLAN	Specify the name of the DB2 plan to be used by the FTP server	562
DCBDSN	Specify a data set to be used as a model for allocation of new data sets	563
DEST	Specify the NJE destination to which the files are routed when you enter a PUT command	565
DIRECTORY	Specify the number of directory blocks to be allocated for the directory of a PDS	566
DIRECTORYMODE	Specify how to treat the data set qualifiers below the current directory	567
FILETYPE	Specify the operational mode of the server	568
INACTIVE	Set the inactivity timer to a specified number of seconds	569
JESLRECL	Specify the record length of the job being submitted	570
JESPUTGETTO	Specify the number of seconds of the JES PutGet time-out	571
JESRECFM	Specify the record format of the job being submitted	572
LRECL	Specify the size of the records in a data set	573
MGMTCLASS	Specify the SMS management class to be assigned to newly allocated data sets	574
MIGRATEVOL	Specify the volume ID for migrated data sets not under the control of IBM storage management systems	575
PRIMARY	Specify the amount of tracks, blocks, or cylinders for primary allocation	576
QUOTESOVERRIDE	Specify use of single quotes in filename	577
RDW	Specify whether RDWs are discarded on retrieval	578
RECFM	Specify the record format of a data set	579
RETPD	Specify the number of days that a newly allocated data set should be retained	580

Table 21. Summary of FTP Server Configuration Statements (continued)

Statement	Description	Page
SBDATACONN	Specify single-byte ASCII/EBCDIC conversion for the data connection	581
SECONDARY	Specify the amount of tracks, blocks, or cylinders for secondary allocation	582
SMF	Specify the default SMF record subtype for all SMF records	583
SMFAPPE	Specify the SMF record subtype for the APPEND subcommand	585
SMFDEL	Specify the SMF record subtype for the DELETE subcommand	586
SMFEXIT	Call the FTPSMFEX user exit routine	587
SMFJES	Collect SMF records when FILETYPE is JES	588
SMFLOGN	Specify the SMF record subtype when recording logon failures	589
SMFREN	Specify the SMF record subtype for the RENAME subcommand	590
SMFRETR	Specify the SMF record subtype for the RETR subcommand	591
SMFSQL	Collect SMF records when FILETYPE is SQL	592
SMFSTOR	Specify the SMF record subtype for the STOR and STOU subcommands	593
SPACETYPE	Specify whether newly allocated data sets are allocated in blocks, cylinders, or tracks	594
SPREAD	Specify output in spreadsheet format when file type is SQL	595
SQLCOL	Specify the column headings of the output file	596
STORCLASS	Specify the SMS-managed storage class for the FTP server	598
TRACE	Start tracing for the FTP server	599
TRAILINGBLANKS	Include trailing blanks in fixed format data sets when retrieved	600
UCSHOSTCS	Specify the EBCDIC code set to be used for data conversion to or from Unicode	601
UCSSUB	Specify whether Unicode-to-EBCDIC conversion should use the EBCDIC substitution character or cause the data transfer to be terminated if a Unicode character cannot be converted to a character in the target EBCDIC code set	602
UCSTRUNC	Specify whether the transfer of Unicode data should be aborted if truncation occurs at the MVS host	603
UMASK	Specify the file mode creation mask.	604
UNITNAME	Specify the unit type for allocation of new data sets	605
VOLUME	Specify the volume serial number for allocation of new data sets	606
WLMCLUSTERNAME	Use the WLMCLUSTERNAME Statment to instruct the FTP Daemon to register in a DNS/WLM Sysplex Connection Balancing Group.	607
WRAPRECORD	Specify whether data will be wrapped or truncated if no new line character is encountered before the logical record length is reached	608
XLATE	Specify the translation table data set for the data connection.	609

Sample FTP Server Configuration Data Set (FTPSDATA)

```
;*********************************************************************
;
;   Name of File:              tcpip.SEZAINST(FTPSDATA)
;
;   Descriptive Name:          FTP.DATA  (for OE-FTP Server)
;
```

```
;   SMP/E Distribution Name:  EZAFTPAS
;
;          5647-A01 (C) Copyright IBM Corp. 1997.
;          Licensed Materials - Property of IBM
;          This product contains "Restricted Materials of IBM"
;          All rights reserved.
;          US Government Users Restricted Rights -
;          Use, duplication or disclosure restricted by
;          GSA ADP Schedule Contract with IBM Corp.
;          See IBM Copyright Instructions.
;
;   This FTP.DATA file is used to specify default file and disk
;   parameters used by the FTP server.
;
;   Note: For an example of an FTP.DATA file for the FTP client,
;   see the FTCDATA example.
;
;   Syntax Rules for the FTP.DATA Configuration File:
;
;   (a) All characters to the right of and including a ; will be
;       treated as a comment.
;
;   (b) Blanks and > are used to delimit tokens.
;
;   (c) The format for each statement is:
;
;       parameter value
;
;***************************************************************************
;
;ANONYMOUS                      ; anonymous login accepted
;ASATRANS      FALSE            ; do NOT translate control characters
                                ; in ASA text
AUTOMOUNT     TRUE             ; automatic mount of unmounted volume
AUTORECALL    TRUE             ; automatic recall of migrated data sets
;AUTOTAPEMOUNT FALSE            ; do NOT automatically mount tape volumes
BLOCKSIZE     6233             ; new data set allocation blocksize
CONDDISP      CATLG            ; data sets catalogued if transfer fails
;CTRLCONN      IBM-850          ; ascii code set for control connection
;DATACLASS     SMSDATA          ; sms data class name
;DB2           D31              ; db2 subsystem name
;DB2PLAN       PLANNAME         ; db2 plan name for OE-FTP
;DCBDSN        MODEL.DCB        ; new data set allocation model dcb name
;DEST          USER14ãMVSL      ; files destination for store
DIRECTORY     27               ; new data set allocation directory blocks
DIRECTORYMODE FALSE            ; directorymode vs. data set mode
FILETYPE      SEQ              ; file transfer mode
INACTIVE      300              ; inactive time out
JESLRECL      80               ; lrecl of jes jobs
JESOUTGETTO   600              ; timeout for remote job submission put/ge
JESRECFM      F                ; recfm of jes jobs
LRECL         256              ; new data set allocation lrecl
;MGMTCLASS     SMSMGMT          ; sms mgmtclass name
;MIGRATEVOL    MIGRAT           ; migration volume volser
PRIMARY       1                ; new data set allocation primary space
;QUOTESOVERRIDE FALSE           ; single quote(s) are treated as part of
                                ; hfs filename, i.e. single quotes do
                                ; NOT indicate working directory override
RECFM         VB               ; new data set allocation record format
;RETPD         30               ; new data set retention period: 30 days
;SBDATACONN   (IBM-1047,IBM-850) ; ebcdic/ascii code sets for data conn.
SECONDARY     1                ; new data set allocation secondary space
;SMF           STD              ; SMF records use standard subtypes
;SMFEXIT                        ; load SMF user exit FTPSMFEX
;SMFJES                         ; SMF recording when filetype=jes
;SMFSQL                         ; SMF recording when filetype=sql
SPACETYPE     TRACK            ; new data set allocation space type
```

```
SPREAD        FALSE          ; sql output format
SQLCOL        NAMES          ; sql output uses column names as headings
;STARTDIR      MVS            ; use MVS directory at connect time
;STORCLASS     SMSSTOR        ; sms storclass name
;TRACE                        ; trace active
;TRAILINGBLANKS TRUE          ; include trailing blanks when fixed
                              ; format data sets are retrieved
;UMASK         027            ; octal UMASK to restrict setting
                              ; of permission bits when creating
                              ; new hfs files
;UNITNAME      3380           ; new data set allocation unit
;VOLUME        WRKLB2         ; new data set allocation volume serial
WRAPRECORD    FALSE          ; data is NOT wrapped to next record
```

Specifying Attributes for New MVS Data Sets

When allocating new data sets, there are two methods you can use to specify the data set attributes. You can individually use the data set attribute parameters with the SITE command or the statements in the FTP.DATA data set. Or, if your system programmer has used the Storage Management System to group together default attributes into named classes, you can specify those class names on the DATACLASS, STORCLASS, and MGMTCLASS statements.

Dynamic Allocation: The FTP server allows a client program to dynamically allocate a new physical sequential data set or a partitioned data set (PDS) for the purpose of transferring data to be written to that data set. The following optional allocation variables can be used to override and turn off the hard-coded defaults that affect the allocation of the data set.

Variable	FTP.DATA statement
allocation units	SPACETYPE
blocksize	BLKSIZE
data class	DATACLASS
directory blocks	DIRECTORY
logical record length	LRECL
management class	MGMTCLASS
model DCB values	DCBDSN
primary space	PRIMARY
secondary space	SECONDARY
record format	RECFM
retention period	RETPD
storage class	STORCLASS
unit	UNITNAME
volume serial number	VOLUME

Some of these allocation variables might provide duplicate information. For example, the model DCB might have a record format (RECFM) that differs from the record format specified by a data class and from the one explicitly specified by the client. FTP passes all variables that are specified to dynamic allocation and lets it determine which of the specifications take precedence. The following list describes the exceptions to that policy:

- If neither the primary nor secondary space quantity is specified, then the allocation units value is not sent.
- If the data set organization is physical sequential, then directory blocks specification is not sent.
- Otherwise, all variables are sent to dynamic allocation where the order of precedence is:

1. Any FTP.DATA statements or SITE parameters explicitly specified or defaulted
2. Any attributes picked up from the model DCB and not otherwise explicitly specified
3. Any attributes picked up from the data class and not previously derived from 1 or 2
4. Any allocation defaults

Storage Management Subsystem (SMS): You can specify one or more of the following SMS classes to manage characteristics that are associated with or assigned to data sets.

- Data class is an SMS construct that determines data set allocation attributes used by SMS for creation of data sets. The fields listed are available attributes that serve as a template for allocation. Each is ***optional*** and is overridden by any explicit specification of FTP allocation variables or by a model DCB (DCBDSN).

Variable	FTP.DATA statement
directory blocks	DIRECTORY
logical record length	LRECL
primary space	PRIMARY
record format	RECFM
retention period	RETPD
secondary space	SECONDARY

  Note: If either primary or secondary space is explicitly specified, then the primary and secondary values from data class are not used.
- Management class is an SMS construct that determines DFHSM action for data set retention, migration, backup, and release of allocated but unused space. Management class replaces and expands attributes that otherwise would be specified. That is, management class might override any other specification of retention period.
- Storage class is a list of storage performance and availability services requests for an SMS-managed data set that SMS attempts to honor when selecting a volume or volumes for the data set. It might conflict with an explicit specification of volume and unit. If storage class is used, then volume and unit should be unspecified.

Step 5: Configure the FTP Server for SMF (Optional)

The FTP server can write type 118 (X'76') SMF records to record transactions made by the FTP server. SMF records are independent of the IP connection. They are created for host connections. SMF records can be written for the following requests:

- APPEND
- DELETE
- RENAME
- RETRIEVE
- STORE
- STORE UNIQUE

Information about the previous requests can be recorded for:

- FTP server running in normal data transfer mode (FILETYPE=SEQ)
- FTP server running remote job submission (FILETYPE=JES)
- FTP server running Structured Query Language (SQL) queries (FILETYPE=SQL)
- Any combination of SEQ, JES, and SQL

For requests involving data transfer (APPEND, GET, PUT, RETR, or STOR) an SMF record will be written for both successfully and unsuccessfully completed data transfer requests which have begun data transfer. For data transfer requests which have completed unsuccessfully, the byte count of transmission field (offset 68) will contain the number of bytes transferred before the failure, and the recent server reply field (offset 73) will contain the 3-digit error reply code sent to the client.

The FTP server can also write SMF records when a logon attempt fails.

The capability also exists for a user-written exit routine to get control before the SMF records are written.

The following sections describe how to configure the FTP server for use with SMF.

Summary of FTP Server SMF Statements

If you want the FTP server to write type 118 (X'76') SMF records, you must include at least one of the SMF subtype statements (SMF, SMFAPPE, SMFDEL, SMFLOGN, SMFREN, SMFRETR, or SMFSTOR) in the FTP.DATA data set.

Table 22 shows the SMF statements in FTP.DATA.

Table 22. Summary of FTP Server SMF Statements

Statement	Description	Page
SMF	Specify the default SMF record subtype for all SMF records	583
SMFAPPE	Specify the SMF record subtype for the APPE (APPEND) subcommand	585
SMFDEL	Specify the SMF record subtype for the DELE (DELETE) subcommand	586
SMFEXIT	Call the FTPSMFEX user exit routine	587
SMFJES	Collect SMF records when FILETYPE is JES	588
SMFLOGN	Specify the SMF record subtype when recording logon failures	589
SMFREN	Specify the SMF record subtype for the RNFR / RNTO (RENAME) subcommand	590
SMFRETR	Specify the SMF record subtype for the RETR (RETRIEVE) subcommand	591
SMFSQL	Collect SMF records when FILETYPE is SQL	592
SMFSTOR	Specify the SMF record subtype for the STOR (STORE) and STOU (STORE UNIQUE) subcommands	593

If SMF subtype statements are not coded in the FTP.DATA data set, then no SMF records are written by the FTP server.

FTP Server SMF User Exit

The FTP server SMF user exit is called before an SMF record that contains information about an FTP server session is written to the SYS1.MANx data set. The user exit allows site specific modifications to the record and controls whether the record is written to the SYS1.MANx data set.

Specify the FTP server SMF user exit option by including the SMFEXIT statement in the FTP.DATA data set. If this option is not specified, the system writes all FTP server SMF records specified in the FTP.DATA data set to the SYS1.MANx data set.

You must name this user exit routine FTPSMFEX, and place it in an installation-defined link library or an APF-authorized data set defined by a STEPLIB DD statement in the FTPD cataloged procedure. FTP calls the SMF user exit before each SMF record is written.

On entry to FTPSMFEX, register 1 contains a pointer to the following 2-word parameter list.

Offset	Value
0	Pointer to the return code
4	Pointer to the SMF record

Prior to calling the SMF user exit, the return code is set to zero. A zero return code specifies that the SMF record will be written. To suppress writing of the SMF record to the SYS1.MANx data set, the user exit must change the return code to a non-zero value.

“Appendix B. SMF Records” on page 1163 contains descriptions of TCP/IP SMF records.

Example FTPSMFEX User Exit

The following example shows an FTPSMFEX user exit.

```
*********************************************************************
* Function:   Allow FTP Server SMF record recording only when       *
*             the client is outside subnet 9.24.104                 *
*********************************************************************
*
FTPSMFEX CSECT
FTPSMFEX AMODE 31
FTPSMFEX RMODE ANY
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         B     BEGIN
         DC    C'FTPSMFEX '
         DC    C' &SYSDATE '
         DC    C' &SYSTIME '
         DS    0F
BEGIN    LR    R2,R1                  *Parm pointer
         USING PARMS,R2
         L     R4,PTRRC               *-> Return code field
         L     R9,PTRSMFR             *-> SMF record
         USING SMFREC,R9
         L     R3,SMFREMIP            *Foreign IP Address
```

```
         SRL   R3,8                    *Get rid of the 8 loworders
         SLL   R3,8                    *Back into line again
         LM    R5,R7,OURBXLE           *Adresses for net loop
NETLOOP  C     R3,0(R5)                *One of our subnets ?
         BE    SKIPREC                 *- Yes, Do not write SMF record
         BXLE  R5,R6,NETLOOP           *Loop through all subnets
         SR    R15,R15                 *Not one of ours - write SMFrec
         B     DONE
SKIPREC  LA    R15,4                   *Do not write record
DONE     ST    R15,0(R4)               *Return the RC
         RETURN (14,12),RC=(15)    RETURN TO CALLER
OURBXLE  DC    A(OURSUB,4,OURSUBSL-4)
OURSUB   DC    AL1(9,24,104,0)                *9.24.104.0 (Production)
OURSUBSL EQU   *
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
PARMS    DSECT
PTRRC    DC    F'0'
PTRSMFR  DC    F'0'
*
SMFREC   DSECT                         *FTP Server SMF record
         DC    24X'00'                 *Std. SMF header
SMFCMD   DC    CL4' '                  *FTP subcommand
SMFFTYPE DC    CL4' '                  *File type (SEQ,JCL,SQL)
SMFREMIP DC    AL4(0)                  *Foreign host IP address
SMFLOCIP DC    AL4(0)                  *Local IP address
         DS    0F                      *Remainder of record not used
*
         END
```

Step 6: Configure the User-Written Exits (Optional)

To limit access to an FTP server, you can use any of the user exits described in this section. The FTP server provides increased security by using 4 user exits. The user exit load modules must be placed in an APF-authorized library which the FTP server has access via STEPLIB, linklist, or LPA. Also, the authorization state (JSCBAUTH) must be the same after exiting from the user exit as it was upon entry. If a user exit is not found, processing proceeds as though a return code of 0 was received from the user exit call.

A user exit is passed the address of a parameter list in register 1. The parameter list is a series of pointers to values. The first word of the parameter list always points to the return code. If the user exit sets the return code to 0, processing continues as normal. If the return code is not 0, authorization is denied and the user receives a negative reply indicating that the command has failed. Upon entry, the return code is 0, so a correct return can be indicated by leaving the return code alone.

The second word of the parameter list always points to a word containing the number of parameters that follow. This helps handle future releases that might increase the number of parameters in these parameter lists.

The remainder of the parameter list points to values that the FTP user exit uses in its processing. Sample user exits are shipped in *hlq*.SEZAINST.

Because the FTPCHKIP user exit is loaded at FTP daemon initialization time, if you want the server to use a new version of your exit routine, you need to recycle the FTP server (stop and start it). If you are debugging a user exit routine, you should have a test version of a server to work with so that you can stop and start without affecting other users. You can do that by putting a PORT parameter in the EXEC statement of the FTP JCL; for example, `PARMS='PORT 1073'`. To connect to this server:

```
FTP nodename 1073
```

You can use any non-well-known number as a port number for your test FTP server.

Note: You cannot use the System Programming C Facilities for the user exits.

The following describes the 4 user exits:

The FTCHKIP User Exit

FTCHKIP is called at the initial stage of logon, or whenever the user issues an OPEN command to open a new connection. The IP and PORT addresses of the local host and remote hosts are passed to the user exit. The user exit can use them to determine if the remote host's control connection should be canceled. The message `421 User Exit rejects open for connection` is sent to the user if the connection is denied. The following parameter list is passed to FTCHKIP.

Offset	Value
+0	Pointer to the word with the return code
+4	Pointer to a word containing the number of following parameters (4)
+8	Pointer to the fullword remote IP address
+12	Pointer to the halfword remote port number
+16	Pointer to the fullword local IP address
+20	Pointer to the halfword local port number

The FTCHKPWD User Exit

FTCHKPWD is called just after the user enters the password. The exit is passed the following information: the user ID, password of the user that has just logged on, and a userdata string. The exit has the option of rejecting the logon. The message `530 User Exit rejects logon by 'xxxxx'` is sent to the user if the logon is denied. The following parameter list is passed to FTCHKPWD.

Offset	Value
+0	Pointer to the word with the return code
+4	Pointer to a word containing the number of following parameters (3)
+8	Pointer to the 8-byte ID of the user logging on

+12 Pointer to the 8-byte password of the user logging on

+16 Pointer to the string containing the 2-byte length field followed by the user data.

The FTCHKCMD User Exit

FTCHKCMD is called whenever the user enters a command to execute (such as GET, PUT, or any other FTP command). The user exit is passed the user ID, the command, and the command parameters. The exit has the option of not permitting the execution of the command. The message `500 User Exit denies Userid xxxxx from using Command yyy` is sent to the user if the command is denied. The following parameter list is passed to FTCHKCMD.

Offset Value

+0 Pointer to the word with the return code

+4 Pointer to a word containing the number of following parameters(3)

+8 Pointer to the 8-byte user ID that is logged on

+12 Pointer to the 8-byte command being entered

+16 Pointer to a string containing arguments after the command. The first halfword of the string contains the number of characters that follow.

The FTCHKJES User Exit

FTCHKJES is called if the server is in FILETYPE=JES mode and the client tries to submit a job. The user ID and the job being submitted are passed to the exit. The exit can allow or refuse the job to be submitted to the JES internal reader. For example, the exit can look for a USER= parameter on the JOB statement and check it against the client's user ID. The message `550 User Exit refuses this job to be submitted by userid` is sent to the user if the remote job submission is denied. The following parameter list is passed to FTCHKJES.

Offset Value

+0 Pointer to the word with the return code

+4 Pointer to a word containing the number of following parameters (8)

+8 Pointer to the 8 character user ID that is logged on

+12 Pointer to the buffer containing the current logical record being submitted

+16 Pointer to a word with the number of bytes in the buffer

+20 Pointer to a word containing the JES LRECL being used

+24 Pointer to a word containing the logical record number

+28 Pointer to a word containing the total number of bytes transferred so far

+32 Pointer to a word containing the unique client ID

+36 Pointer to a word containing the JES RECFM (0 for fixed, 1 for variable)

+40 Pointer to a word containing the JES user exit anchor. (One possible use of this anchor is to provide the exit routine with a location to store the address of a persistent storage area for handling multiple calls.)

The return code word is initialized to 0, so the user exit can return without changing it if there is a correct return code. Any other return code denies access to the resource in question.

Notes:

1. FTCHKIP has been placed before the user logs on, and if access is denied, the user receives a message and then the control connection is severed. This message comes at a point when most clients expect to continue with the logon process by sending the user ID and password. Even though it is possible that some FTP clients might not expect a 421 message at this point, it is the most appropriate place for this exit.
2. MVS follows the MVS search order to load the FTP exit routines. If you are not using the user exit facility, put a dummy user exit load module in the first library in the MVS search order. This prevents someone from putting in their own module in a library later in the concatenation sequence. This also increases the need to have that library protected using SAF.

Step 7: Specify Configuration Statements in TCPIP.DATA

The FTP server gets certain operating parameters from the statements in the TCPIP.DATA data set. This data set has statements that set the TCP/IP client system parameters. Table 23 shows the statements which specifically affect the FTP server. For the search order used for the TCPIP.DATA data set, see “Configuration Data Sets” on page 53.

Summary of FTP Server TCPIP.DATA Statements

Table 23. Summary of FTP Server TCPIP.DATA Statements

Statement	Description	Page
DATASETPREFIX	Set the default high-level qualifier for configuration data sets	293
DOMAINORIGIN	Specify the domain origin that is appended to the host name to form the fully qualified domain name for a host	294
HOSTNAME	Specify the TCP host name of the OS/390 server	295
LOADDBCSTABLES	Tell FTP which DBCS translation tables can be loaded	296
MESSAGECASE	Specify case translation for the FTP server, PING client, oping, onetstat, orouted, and osnmpd	298

The following is an example of the statements that affect FTP in the TCPIP.DATA data set:

```
HostName MVS1
DOMAINORIGIN IDD.RALEIGH.IBM.COM
LOADDBCSTABLES TCHINESE
MESSAGECASE LOWER
DATASETPREFIX TCP.PROD
```

See “Chapter 6. Defining the TCP/IP Client System Parameters” on page 285 for detailed information about the TCPIP.DATA data set.

Step 8: Install the SQL Query Function (Optional) and Access the DB2 Modules

To use FTP to do SQL queries, bind the DBRM called EZAFTPMQ to the plan used by FTP, and grant execution privileges for that plan to PUBLIC. (The name of the plan can be specified by the DB2PLAN keyword in FTP.DATA or defaulted to EZAFTPMQ.) This FTP facility only performs SELECT operations on the DB2 tables. It does not perform UPDATE, INSERT, or DELETE.

Note: If secondary authorization for SQL queries is required, the DSN3SATH sample exit shipped by DB2 must be modified. The exit will return the primary authid for requests originating from the FTP server.

The following sample job is provided in the FTOEBIND member of the SEZAINST data set. It can be used to enable the FTP server and client to do SQL queries.

```
//FTPSETUP JOB FTPSETUP,
//            CLASS=A,
//            NOTIFY=&SYSUID
//*************************************************************
//*
//*   File name:                 tcpip.SEZAINST(FTOEBIND)
//*   SMP/E distribution name:   EZAFTPAB
//*
//*    Licensed Materials - Property of IBM
//*    This product contains "Restricted Materials of IBM"
//*    5647-A01 (C) Copyright IBM Corp.  1997.
//*    All rights reserved.
//*    US Government Users Restricted Rights -
//*    Use, duplication or disclosure restricted by GSA ADP Schedule
//*    Contract with IBM Corp.
//*    See IBM Copyright Instructions.
//*
//*  This JCL binds the EZAFTPMQ DBRM to the specified
//*  DB2 subsystem and allows execution of the
//*  EZAFTPMQ plan by PUBLIC.
//*
//*  The MVS OE FTP server and client use this plan. (See
//*  Usage note #7)
//*
//*
//*  Usage notes:
//*
//*    1.  You must execute this job from a user ID that has
//*        the authority to bind the EZAFTPMQ plan.
//*
//*    2.  Change the STEPLIB DD statement in the FTPBIND and
//*        FTPGRANT steps to reflect the DB2 DSNLOAD data set.
//*
//*    3.  Change the DB2 sybsystem name in the FTPBIND and
//*        FTPGRANT steps from SYSTEM(xxx) to the
//*        installation defined DB2 subsystem name.
//*
//*    4.  Change the library parameter in the FTPBIND step from
//*        TCPIP.SEZADBRM to the installation defined TCPIP
//*        SEZADBRM library.
//*
//*    5.  Change the plan name in the FTPGRANT step from
//*        DSNTIAYY to reflect the plan associated with the
//*        program DSNTIAD.
//*
//*    6.  Change the library parameter in the FTPGRANT step
//*        from xxxxxx.RUNLIB.LOAD to reflect the library
//*        where the DSNTIAD program resides.
//*
//*    7.  You can bind the DBRM to a plan name other than EZAFTPMQ
//*        by changing the plan specified in the FTPBIND and
//*        FTPGRANT steps.  If you do this, you must use the
//*        DB2PLAN keyword in FTP.DATA to change the plan name
//*        used by the FTP server and/or client to the plan name
//*        specified here.
//*
//*************************************************************
//FTPBIND  EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
```

```
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 BIND ACQUIRE(USE) -
      ACTION(REPLACE) -
      CACHESIZE(1024) -
      CURRENTDATA(NO) -
      EXPLAIN(NO) -
      ISOLATION(CS) -
      LIBRARY('TCPIP.SEZADBRM') -
      MEMBER(EZAFTPMQ) -
      NODEFER(PREPARE) -
      PLAN(EZAFTPMQ) -
      RELEASE(COMMIT) -
      VALIDATE(RUN) -
      RETAIN
 END
//*
//FTPGRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 RUN  PROGRAM(DSNTIAD) -
      PLAN(DSNTIAYY) -
      LIBRARY('xxxxxx.RUNLIB.LOAD')
 END
//SYSIN    DD   *
 GRANT EXECUTE ON PLAN EZAFTPMQ TO PUBLIC;
//*
```

Accessing DB2 Modules

The FTP server or client loads 3 DB2 modules into storage to perform an SQL query. These modules are:

- DSNALI
- DSNHLI2
- DSNTIAR

The modules are usually found in the DB2 load library with the suffix DSNLOAD. The DB2 administrator or system programmer should add the DSNLOAD library to the LINKLIST to ensure that FTP has access to this library.

Another way to ensure access is to add the DSNLOAD library to the FTP STEPLIB. For the FTP server this means that the JCL that is used to start the FTP server has a STEPLIB DD statement referring to the DSNLOAD library or, if the FTP daemon is started from the OS/390 shell, that the STEPLIB environment variable is set. For the FTP client, this means that a TSO CLIST must allocate the DSNLOAD library as the STEPLIB.

If the FTP client is to be run from a batch job to perform SQL queries, the DSNLOAD library must be added to the STEPLIB DD statement for the batch job.

Usage Notes:

To allow FTP access to multiple levels of DB2, link to the libraries that contain the lowest level of DB2 to be accessed.

Step 9: Update /etc/syslog.conf for the FTP Server

The daemon.*priority* entries in /etc/syslog.conf determine where FTP messages and trace entries are written. The FTP server issues info, warning, and error messages. All trace entries are written with debug priority. To direct trace entries (and all messages) to /tmp/daemon.trace, include the following in /etc/syslog.conf:

```
daemon.debug    /tmp/daemon.trace
```

For information about syslogd, see “Appendix E. Syslog Daemon” on page 1183.

Implementing an Anonymous User

Using an FTPD server requires certain security considerations. It is common to provide FTP services for unidentified users; this is the so-called ANONYMOUS user that logs in with a user ID of ANONYMOUS. The OpenEdition FTPD server supports both fully identified users and the anonymous user.

Anonymous user support is a configuration option. If you implement it, you must specify how the support is going to be implemented in your environment.

Identified User

The FTP daemon is a never-ending server process that waits for connection requests on the well-known port number 21. After an FTP client connects, the FTPD listener process starts another process that will be used for this individual FTP session. In this setup, the FTP daemon process was started with a user ID of TCPIP3 as shown in Figure 22.

```
D OMVS,A=ALL
BPXO040I 17.26.55 DISPLAY OMVS 731
OMVS     000E ACTIVE          OMVS=(03)
USER     JOBNAME ASID         PID      PPID  STATE  START     CT_SECS
OMVSKERN BPXOINIT 0025          1         0  MKI    15.51.47   .089
   LATCHWAITPID=        0 CMD=BPXPINPR
   SERVER=Init Process                   AF=      0 MF=65535 TYPE=FILE
TCPIP3   T03ATCP   00FC  16777218         1 MR      15.56.35   59.391
LATCHWAITPID=           0 CMD=EZBTCPIP
TCPIP3   T03ATCP   00FC  16777219         1 1F      15.56.40   59.391
LATCHWAITPID=           0 CMD=EZACFALG
TCPIP3   T03ATCP   00FC 100663300         1 1F      15.56.42   59.391
LATCHWAITPID=           0 CMD=EZASASUB
TCPIP3   T03ATCP   00FC         5         1 1R      15.56.51   59.391
LATCHWAITPID=           0 CMD=EZBTMCTL
TCPIP3   T03ATCP   00FC         6         1 1R      15.56.53   59.391
LATCHWAITPID=           0 CMD=EZBTTMST
TCPIP3   T03FTP3 2  0036  167772169  50331660 1FI   17.26.43   .232
LATCHWAITPID=           0 CMD=/usr/sbin/ftpdns 6233 0 27 1 80 128 256
TCPIP3   T03FTP1 1 001C   50331660        1 1FI     17.07.05   1.040
LATCHWAITPID=           0 CMD=FTPD
```

Figure 22. FTPD Server Before User Enters Login Information

In the example above, **1** and **2** respectively indicate:

1. The FTPD listener process that executes the ftpd program.

2. The new process that handles the new client connection. The program in this process is the ftpdns program. At this time, the FTP client user has not yet entered a user ID and password, which is the reason why this process still executes under the same user ID as the FTP daemon process.

Figure 23 shows the server after the user has entered the login information.

```
D OMVS,A=ALL
BPXO040I   17.29.09 DISPLAY OMVS 737
OMVS       000E ACTIVE          OMVS=(03)
USER       JOBNAME  ASID        PID      PPID STATE  START     CT_SECS
OMVSKERN BPXOINIT 0025           1        0 MKI     15.51.47  .089
   LATCHWAITPID=            0 CMD=BPXPINPR
   SERVER=Init Process                   AF=     0  MF=65535 TYPE=FILE
TCPIP3     T03ATCP 00FC     16777218       1 MR      15.56.35  60.603
LATCHWAITPID=               0 CMD=EZBTCPIP
TCPIP3     T03ATCP 00FC     16777219       1 1F      15.56.40  60.603
LATCHWAITPID=               0 CMD=EZACFALG
TCPIP3     T03ATCP 00FC     100663300      1 1F      15.56.42  60.603
LATCHWAITPID=               0 CMD=EZASASUB
TCPIP3     T03ATCP 00FC             5      1 1R      15.56.51  60.603
LATCHWAITPID=               0 CMD=EZBTMCTL
TCPIP3     T03ATCP 00FC             6      1 1R      15.56.53  60.603
LATCHWAITPID=               0 CMD=EZBTTMST
KAKKY T03FTP3 1 0036        167772169 50331660 1FI   17.26.43  .285
LATCHWAITPID=               0 CMD=/usr/sbin/ftpdns 6233 0 27 1 80 128 256
TCPIP3 T03FTP1 2   001C     50331660       1 1FI     17.07.05  1.040
LATCHWAITPID=               0 CMD=FTPD
```

Figure 23. FTPD Server After User Enters Login Information

In the example above, **1** and **2** respectively indicate:

1. After the user enters a user ID and password, the process changes its user ID.
2. Again, you can verify the relationship between parent and child processes by picking up the PPID of the user process and the PID of the server process.

   Note: All users of the OS/390 V2R8 IP FTPD server must have a valid OMVS segment in their RACF user profiles. This is true even when the user only wants to access MVS data sets via the OS/390 V2R8 IP FTPD server.

If the FTP client end user maneuvers in a way that is not allowed, the file system and RACF immediately notice it, the request fails, and a corresponding RACF message is issued on the MVS console as shown in Figure 24.

```
ICH408I USER(KAKKY ) GROUP(WTCRES ) NAME(TATSUHIKO KAKIMOTO )
  /etc/orouted.03a.env
  CL(FSOBJ     ) FID(01E2D7D3C5E7F34E2B0D0000106A0000)
  INSUFFICIENT AUTHORITY TO OPEN
  ACCESS INTENT(R--) ACCESS ALLOWED(OTHER ---)
```

Figure 24. FTP User Attempt to Violate Access Authorization

Notes:

1. CL(FSOBJ) identifies an internal RACF class. It refers to a file system object. You do not need to define it anywhere; it is used for auditing purposes. You

might see the following classes in similar messages: DIRACC, DIRSRCH, FSSEC, IPCOBJ, PROCACT, PROCESS.

2. FID identifies the file system object in question.

Anonymous User

A special situation with respect to user identification occurs with FTP. This service might be accessed by a so-called ANONYMOUS user. The OpenEdition FTPD server must be specifically configured to support anonymous access. The default option is to refuse anonymous logon. The anonymous support can be enabled in three different ways. The definitions are made in the FTPD server's FTP.DATA configuration data set or file. The syntax for defining anonymous access is as follows:

Parameter
Description

No options
When an FTP client enters a user ID of anonymous, the FTPD server uses an MVS user ID of ANONYMO. The user is prompted for a valid password. If this user ID has not been defined in RACF or if the user ID has been defined but no OMVS segment created, the logon request fails.

user_id
When the FTP client user enters a user ID of anonymous, the FTPD server acts as if the user had entered the specified user ID and the user is prompted to enter the correct password. All access to resources will be made based on the specified user ID. If this user ID has not been defined in RACF or if the user ID has been defined but no OMVS segment created, the logon request fails.

user_id/password
When the FTP client user enters a user ID of anonymous, the FTPD server will act as if the user had entered the specified user ID, but the user will not be requested to enter a password. The FTPD server will use the specified password in its request to RACF to create a user security environment.

As noted in the previous description, it follows immediately that support of anonymous users is only possible in the last case. This means you must add an anonymous definition to your FTP.DATA configuration data set or file as shown in the following example:

```
ANONYMOUS userid/password
```

In our sample setup, a user ID called FTPANOM was defined as follows:

```
ADDUSER FTPANOM OMVS(UID(999) HOME(/u/ftpanom) PGM(/bin/sh))
```

To limit access to resources in the hierarchical file system, this user was connected to the same RACF group used for public access to the MVS Web server, the EXTERNAL group.

Because user FTPANOM has no TSO segment defined, if someone were able to read FTP.DATA where the password has to be stored in clear text, no TSO access

is possible. If the password is compromised, another possible entry into MVS could be through rlogin or telnet. To manage that situation, a profile was defined in the /u/ftpanom directory that just contained an exit command a shown the following example:

```
echo Hello $LOGNAME.
echo Sorry, you do not have access to the system.
exit
```

Furthermore in this example, the ownership of .profile was removed, OMVSKERN was assigned as the owner, and the attribute modified to 045, which restricts FTPANOM to just reading and executing this file. To write a shell history log, .sh_history has attributes of 060 in order that FTPANOM can write to it.

```
pwd=/u/ftpanom: >ls -nal
total 24
drwx------ 2   999    999  0 May 7  01:18 .
drwxr-xr-x 13  0      0    0 May 7  00:58 ..
----r--r-x 1   0      999  69 May 7 01:08 .profile
----rw---- 1   999    999  14 May 7 01:48 .sh_history
-rw-r----- 1   999    999  57 May 7 01:17 test.scr
```

A sample FTP logon sequence looks like the following example:

```
[C:\]ftp mvs18o
IBM TCP/IP for OS/2 - FTP Client ver 08:36:08 on Jul 22 1996
Connected to mvs25.ITSO.RAL.IBM.COM..
220-T25OFTP1 IBM MVS V3R3 at mvs25itso.ral.ibm.com, 17:44:08 on 1997-08-28.
220 Connection will close if idle for more than 5 minutes.
Name (mvs18o): anonymous
230 'ANONYMOUS' logged on. Working directory is "/ u/ftpanom" .
ftp> dir
200 Port request OK.
125 List started OK
total 8
-rw-r----- 1 FTPANOM EXTERNAL 57 May 6 23:17 test.scr
250 List completed successfully.
73 bytes received in 0.09 seconds (0 Kbytes/s)
ftp>
```

Security Considerations for the FTP Server

Consider the following for security:

- User IDs

  To log into the FTP server, a user ID must have an OS/390 UNIX uid.
- The FTPD cataloged procedure must be:
  - Defined to the security program
  - Added to the RACF started class facility or the started procedures table. The userid associated with the FTP server started class must have a UID of 0.
- Terminal Access

  The terminal ID passed from FTP to RACF is an 8-byte hexadecimal character string containing an IP address. RACF interprets this as a terminal logon address and rejects it if it is not previously defined. For example, the IP address 163.97.227.17 is translated to X'A361E311'.

  Therefore, if the SETROPTS TERMINAL(NONE) setting is used in RACF, you must define profiles for the IP addresses in class TERMINAL to avoid problems when trying to FTP to MVS. You must translate all the IP addresses of any clients connecting to FTP servers to hexadecimal character strings and add them to the class TERMINAL.

To allow access by all addresses starting with “163,” define a profile for all addresses in the 163.97.227 subnet:

```
RDEFINE TERMINAL A361E3* UACC(READ)
```

If your RACF SETROPTS options are TERMINAL(READ), all terminals are allowed access to your system, and you do not have to add extra resource definitions to your RACF data base.

For more information, see *OS/390 UNIX System Services Planning* and the *OS/390 Security Server (RACF) Security Administrator’s Guide*.

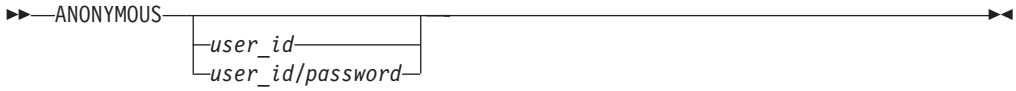
FTP.DATA Data Set Statements

These sections cover, in detail, the statements you can use in the FTP.DATA data set.

ANONYMOUS Statement

Use the ANONYMOUS statement to allow a remote user to issue USER ANONYMOUS without supplying a logon password. Specify ANONYMOUS to make your universally permitted data sets accessible to all users on the TCP/IP network.

Syntax



Parameters

user_id
The name of the user ID to be used when ANONYMOUS is entered as the user ID. When a remote user enters ANONYMOUS as a user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The user will be prompted for the password to *user_id* and, if the user enters the correct password, the user will be logged in as the specified *user_id*.

If you are using RACF, the system will build a user Accessor Environment Element (ACEE) and the ANONYMOUS user will have access to any resources available to the specified user ID.

user_id/password
The name of the user ID and password to be used when ANONYMOUS is entered as a user ID. When a remote user enters ANONYMOUS as the user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The FTP server automatically provides the *password* for the specified *user_id* and the user will be logged in as the specified *user_id*. If you are using RACF, the system will build the user ACEE for the specified *user_id* and the ANONYMOUS user will have authorized access to the same resources as the specified *user_id*.

Examples

Allow a remote user to enter ANONYMOUS as a user ID and be connected to the server system with the user ID of TERMABC:

```
ANONYMOUS TERMABC/ILLBBACK
```

Usage Notes

- If you specify ANONYMOUS without a userid:
  - The userid ANONYMO must be defined and must have an OS/390 UNIX segment defined or defaulted.
  - The end user will not be prompted for a password.
  - If you are using the FTCHKPWD user exit, it will be called with userid = 'ANONYMO' and password = '*'.
  - The initial working directory will be 'ANONYMO.' or the home directory for the ANONYMO userid, depending on the setting of the STARTDIRECTORY FTP.DATA statement.

  - If you are using RACF, a user who logs in as 'anonymous' will have access to any resources that are accessible to the ANONYMO userid.
- If you specify a userid on the ANONYMOUS statement, that userid must be defined and have an OS/390 UNIX segment defined or defaulted.
- There is no default for ANONYMOUS. If you do not include the ANONYMOUS statement in FTP.DATA, then anonymous user login will not be allowed.

ASATRANS Statement

Use the ASATRANS statement to control how the server handles ASA file transfers. Choose either to have the control characters converted by the C runtime during a file transfer or to transfer them without conversion.

The conversion process is described in the *IBM C/370 Programming Guide* in the chapter on ASA Text Files.

Syntax

Parameters

TRUE
Characters in column 1 of the file being transferred are converted to C control character sequences.

FALSE
Characters in column 1 of the file being transferred are not converted. This is the default.

Examples

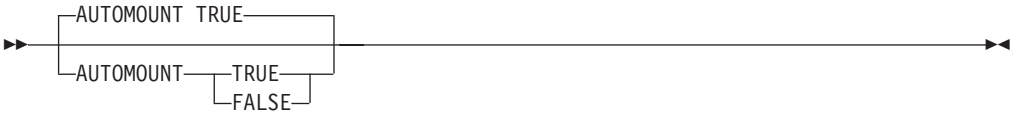
Convert characters in column 1 of the file being transferred:

```
ASATRANS TRUE
```

AUTOMOUNT Statement

Use the AUTOMOUNT statement to permit DASD volumes that are not mounted to be automatically mounted.

Syntax



Parameters

TRUE
Permits DASD volumes that are not mounted to be automatically mounted. This is the default.

FALSE
Prevents DASD volumes that are not mounted from being automatically mounted.

Examples

Mount DASD volumes that are not already mounted automatically:

```
AUTOMOUNT TRUE
```

AUTORECALL Statement

Use the AUTORECALL statement to specify whether data sets which have been migrated by a storage manager, such as HSM, will be recalled automatically.

Syntax

Parameters

TRUE
Permits data sets migrated by the storage manager, such as HSM, to be recalled automatically. This is the default.

FALSE
Prevents migrated data sets from being recalled automatically.

Examples

Recall migrated HSM files automatically:

```
AUTORECALL TRUE
```

Usage Notes

- Migrated data sets can still be deleted even though you specify `FALSE`.
- Partitioned data set members require the whole data set to be recalled.

AUTOTAPEMOUNT Statement

Use the AUTOTAPEMOUNT statement to specify whether tapes that are not already mounted are to be automatically allocated and mounted.

Syntax

Parameters

TRUE
Permits tapes that are not mounted to be automatically allocated and mounted. This is the default.

FALSE
Prevents tapes that are not mounted from being automatically allocated and mounted.

Examples

Automatically mount tape volumes that are not already mounted:

```
AUTOTAPEMOUNT TRUE
```

Do not automatically mount tape volumes that are not already mounted.

```
AUTOTAPEMOUNT FALSE
```

BLKSIZE Statement

Use the BLKSIZE statement to specify the block size of newly allocated data sets.

Syntax

Parameters

size
Specifies the block size of newly allocated data sets. The valid range is 0 through 32760. Specifying no value for block size allows the block size from a model DCB data set or SMS dataclass to be used. The default block size is 6233.

Examples

Set block size to 6144 bytes:

```
BLKSIZE 6144
```

Specify no value for blocksize to allow the blocksize from a model DCB data set or SMS dataclass to be used:

```
BLKSIZE
```

Usage Notes

- If you specify the BLKSIZE statement without a *size*, FTP will not specify the block size when allocating new data sets.
- You should use the BLKSIZE statement without a *size* if you have specified the DATACLASS statement and the block size from the SMS data class is to be used.
- You can also specify this statement as BLOCKSIZE.

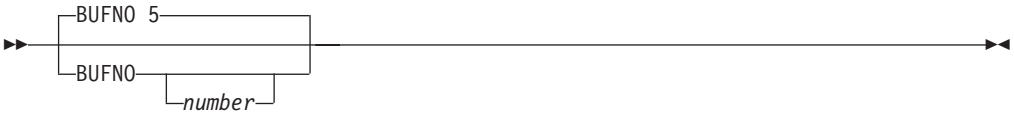
Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “DCBDSN Statement” on page 563

BUFNO Statement

Use the BUFNO statement to specify the number of access method buffers that are used when data is read from or written to a data set.

Syntax



Parameters

number
Specifies the number of buffers allocated. The valid range is 1 through 35. The default is 5.

CCXLATE Statement

Use the CCXLATE statement to specify a data set containing translate tables to be used for the control connection.

Syntax

Parameters

name
Specifies a 1-8 character name that corresponds to a data set that contains translate tables.

FTP looks first for an environment variable called _FTPXLATE_*name*. If the environment variable exists, its value is used as the data set name.

Note: The environment variable name must be all uppercase, although the CCXLATE parameter can be in mixed case.

If the environment variable does not exist FTP looks for a data set called *hlq*.*name*.TCPXLBIN.

Examples

```
CCXLATE FRED
```

If environment variable _FTPXLATE_FRED=FREDDYS.TABLES is defined for the FTP server, the statement above specifies that the translate tables in data set FREDDYS.TABLES should be used for the control connection.

If there is no such environment variable defined, the above statement specifies that the translate tables data set *hlq*.FRED.TCPXLBIN should be used.

Usage Notes

- CCXLATE and CTRLCONN are mutually exclusive statements. If both statements appear in your FTP.DATA file, CCXLATE will be ignored.
- The CCXLATE statement (and its value) is not case-sensitive, but the name of the corresponding environment variable must be all uppercase or FTP will not recognize it.

Related Topics

- “Defining Optional Environment Variables for the FTP Server” on page 527
- “XLATE Statement” on page 609
- To see the search order that determines the conversion for the control connection, see “FTP Code Page Conversion” on page 613.

CHKPTINT Statement

Use the CHKPTINT statement to specify the checkpoint interval when the FTP server is the sending site in a file transfer request.

Syntax

Parameters

number
Used to determine when a restart marker is transmitted. The marker is transmitted after the specified number of records are sent.

If *number* is set to zero, then no checkpointing occurs and no marker blocks are transmitted. The default is zero.

Examples

To send a restart marker of every 100000 records:

```
CHKPTINT 100000
```

Usage Notes

If a nonzero value is coded for CHKPTINT, checkpoint markers are sent to each client who uses EBCDIC blockmode or EBCDIC compress mode during data set retrieval. If some of those clients do not support the restart marker, you can accept the default value of 0 on this statement and, instead, set the checkpoint interval for an individual client with the FTP SITE command. See *OS/390 SecureWay Communications Server: IP User's Guide* for more information on the SITE command.

CONDDISP Statement

Use the CONDDISP statement to keep and catalog or delete a data set when an FTP file transfer ends prematurely.

Syntax

Parameters

CATLG
Specifies that a data set is kept and cataloged when an FTP file transfer ends prematurely. This is the default.

DELETE
Specifies that a data sets is deleted when a file transfer ends prematurely.

Examples

Specify that a data set is deleted when a file transfer ends prematurely:

```
CONDDISP DELETE
```

Usage Notes

- DELETE is ignored if the file transfer ended prematurely because the FTP server was stopped.
- DELETE is ignored if the server receives a checkpoint marker.

CTRLCONN Statement

This statement defines the ASCII code page to be used for the control connection.

Syntax

Notes:

1 7BIT is the default if CTRLCONN is not used and no TCPXLBIN data set is found.

Parameters

7BIT
Indicates 7-bit ASCII is to be used

iconv_ascii
A name recognized by iconv to indicate an ASCII code page.

Examples

```
CTRLCONN IBM-850
```

Usage Notes

7BIT or an *iconv_ascii* name can be entered in lowercase or uppercase.

To see the search order that determines the code page conversion for the control connection, see “FTP Code Page Conversion” on page 613.

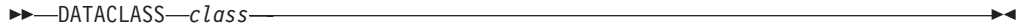
Related Topics

For the code pages supported, see code set converters in the *OS/390 C/C++ Programming Guide*.

DATACLASS Statement

Use the DATACLASS statement to specify the SMS-managed data class as defined by your organization for the FTP server.

Syntax



Parameters

class
The SMS-managed data class as defined by your organization. There is no default.

Examples

Use the SMS data class SMSDATA when allocating new data sets:

```
DATACLASS SMSDATA
```

Usage Notes

If you specify any of the following FTP.DATA statements, or let them default, the values specified or defaulted will override the value specified in the SMS DATACLASS:

- BLKSIZE
- DIRECTORY
- LRECL
- PRIMARY
- RECFM
- RETPD
- SECONDARY

If you specify the DCBDSN statement, the LRECL, RECFM, BLKSIZE, and RETPD (if specified) of the DCBDSN data set will override the values specified in the SMS DATACLASS. To prevent these keywords from overriding the values specified in the SMS DATACLASS, specify them with no keyword values.

If you specify the MGMTCLASS statement, and the requested management class specifies a retention period, the RETPD value of the management class might override the RETPD value of DATACLASS.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “BLKSIZE Statement” on page 553
- “DCBDSN Statement” on page 563
- “DIRECTORY Statement” on page 566
- “LRECL Statement” on page 573
- “MGMTCLASS Statement” on page 574

- “PRIMARY Statement” on page 576
- “RECFM Statement” on page 579
- “RETPD Statement” on page 580
- “SECONDARY Statement” on page 582

DB2 Statement

Use the DB2 statement to specify the name of the DB2 subsystem.

Syntax

Parameters

subsystem_name
The name of the DB2 subsystem. The default name is DB2.

Examples

Set the DB2 subsystem name to DB2X:

```
DB2 DB2X
```

DB2PLAN Statement

Use the DB2PLAN statement to specify the DB2 plan to be used by the FTP server.

Syntax

Parameters

plan_name
The name of the DB2 plan bound in the DB2 subsystem.

Examples

Set the plan name to FTPPLAN:

```
DB2PLAN    FTPPLAN
```

DCBDSN Statement

Use the DCBDSN statement to specify an MVS data set to be used as a model for allocation of new data sets.

Syntax

Parameters

name
The name of the data set that will be used as a model for allocation of new data sets created with a STOR or MKDIR command. This data set name must be a fully-qualified MVS data set name; HFS file names are not allowed. There is no default.

Examples

Use model.dcb as the model data set for allocation and specify RECFM, LRECL, and BLKSIZE with no parameters to allow the attributes from the model DCB to be used:

```
DCBDSN model.dcb
BLKSIZE
LRECL
RECFM
```

BLKSIZE and LRECL can also be specified with a value of 0 to allow the attributes from the model DCB to be used:

```
DCBDSN model.dcb
BLKSIZE 0
LRECL 0
RECFM
```

Usage Notes

If specified or defaulted, the following FTP.DATA statements or SITE command parameters will override the DCB values from the model data set:

- BLKSIZE
- LRECL
- RECFM
- RETPD

If you specify the MGMTCLASS statement, the retention period from the model data set can be overridden by the retention period specified by the SMS management class.

When using a model DCB at the server, SENDSITE must be toggled off at the client. Otherwise, the SITE information that is sent automatically by the client will override the value provided by the model DCB.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “BLKSIZE Statement” on page 553
- “LRECL Statement” on page 573
- “MGMTCLASS Statement” on page 574
- “RECFM Statement” on page 579
- “RETPD Statement” on page 580

DEST Statement

Use the DEST statement to specify the NJE destination to which the files are routed when you enter a STOR command. Using the DEST statement allows you to send data sets to other users on machines that are connected on a Network Job Entry (NJE) network rather than storing them at the server.

Syntax

Parameters

destination
The NJE destination to which the files are routed when you enter a PUT command. The format for *destination* should be one of the following:

- userID@nodeID
- nodeID.userID
- nodeID
- DestID

There is no default.

Examples

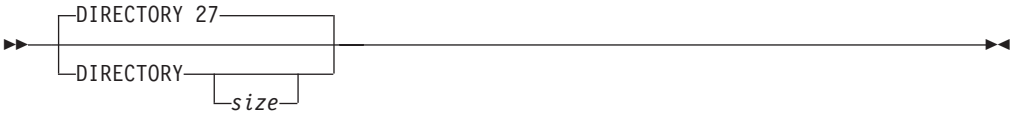
Send files to user USER14 at system MVS1 on a STOR command:

```
DEST USER14@MVS1
```

DIRECTORY Statement

Use the DIRECTORY statement to specify the number of directory blocks to be allocated for the directory of a PDS.

Syntax



Parameters

size
The number of directory blocks to be allocated for the directory of a PDS. The valid range is 1 to 16777215 blocks (the operating system maximum). The default is 27.

Examples

Allocate a PDS with 15 directory blocks:

```
Directory 15
```

Specify DIRECTORY with no value to allow the directory information from an SMS dataclass to be used:

```
DIRECTORY
```

Usage Notes

- If you specify no value for the *size*, FTP will not specify the number of directory blocks to be allocated for the directory of a PDS.
- You should specify no value for the *size* if the DATACLASS statement is specified and the directory from the SMS data class is to be used.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559

DIRECTORYMODE Statement

Use the DIRECTORYMODE statement to specify whether only the data set qualifier immediately below the current directory is treated as an entry in the directory or if all the data set qualifiers below the current directory are treated as entries in the directory.

Syntax

Parameters

TRUE
Specifies that only the data set qualifier immediately below the current directory is treated as an entry in the directory.

FALSE
Specifies that all the data set qualifiers below the current directory are treated as entries in the directory. This is the default.

Examples

Use all qualifiers (Datasetmode):

```
DirectoryMode FALSE
```

Usage Notes

In directory mode, only the data set qualifier immediately below the current directory is the only one used by the MPUT, MGET, LS, and DIR subcommands.

FILETYPE Statement

Use the FILETYPE statement to specify the mode of operation of the server.

Syntax

Parameters

JES
Remote job submission

SEQ
Sequential or partitioned data sets. This is the default.

SQL
SQL query function

Examples

Set the operational mode to SQL:

```
Filetype SQL
```

INACTIVE Statement

Use the INACTIVE statement to set the inactivity timer to a specified number of seconds. Any client control connection which is inactive for the amount of time specified on this statement is closed by the server.

Syntax

Parameters

seconds
The number of seconds to which the inactivity timer will be set. The valid range is 0 through 86400. The default is 300.

Examples

Set the inactivity timer to 30 seconds:

```
INACTIVE 30
```

Usage Notes

If you specify 0 seconds, the inactivity timer will be disabled and the control connections will never time out. This value has no effect on the data connections. To specify a timeout value for the data connection, use the KEEPALIVEOPTIONS statement in TCPIP.DATA. Refer to “Step 1: Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS Information” on page 523 for details.

JESLRECL Statement

Use the JESLRECL statement to specify the record length of the jobs being submitted.

Syntax

Parameters

length
The record length of the job being submitted. The valid range is 1–254. The default is 80. If you specify *length* as *, FTP uses the length value from the LRECL statement.

Examples

Explicitly set the logical record length for JES jobs to 80:

```
JESLRECL 80
```

JESOUTGETTO Statement

Use the JESOUTGETTO statement to specify the number of seconds of the JES PutGet time-out.

Syntax

Parameters

seconds
The number of seconds of the JES PutGet time-out. The valid range is 0 through 86400 (24 hours). The default is 600 (10 minutes).

Examples

Set the number of seconds of the JES PutGet time-out to 300:

```
JESPUTGETTO 300
```

JESRECFM Statement

Use the JESRECFM statement to specify the record format of the jobs being submitted.

Syntax

Parameters

F Fixed record length. This is the default.

V Uses the record format specified on the RECFM statement.

***** Uses the record format specified on the RECFM statement.

Examples

Use fixed record format:

```
JESRECFM F
```

Usage Notes

Only use the value V when running on JES3 systems.

LRECL Statement

Use the LRECL statement to specify the size of the records in a data set.

Syntax

Parameters

length
The size of the records in a data set. The valid range is 0 through 32760. The default is 256.

Examples

Set the logical record length to 128 bytes:

```
LRECL 128
```

Specify no value for LRECL to allow the LRECL of a model DCB data set or SMS dataclass to be used:

```
LRECL
```

Usage Notes

- If you specify no value for *length*, FTP will not specify the size of the records in a data set.
- You should specify no value for *length* if the DATACLASS statement is specified and the LRECL from the SMS data class is to be used, or if the DCBDSN statement is specified and the LRECL from the model data set is to be used.

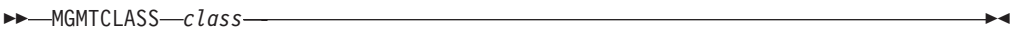
Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “DCBDSN Statement” on page 563

MGMTCLASS Statement

Use the MGMTCLASS statement to specify the SMS management class to be assigned to newly allocated data sets.

Syntax



Parameters

class
The SMS management class.

Examples

Set the SMS management class for new data sets to TCPMGMT:

```
MGMTCLASS TCPMGMT
```

Related Topics

See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.

MIGRATEVOL Statement

Use the MIGRATEVOL statement to specify the volume ID for migrated data sets under the control of a storage management system other than HSM.

Syntax

Parameters

volume_id
The volume ID for migrated data sets. The default volume ID is MIGRAT.

Examples

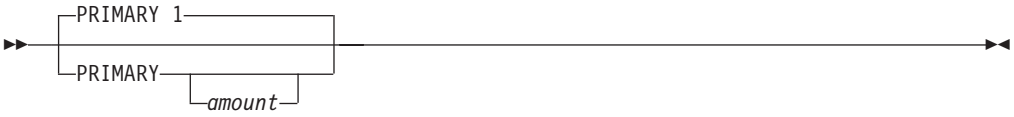
Set the volume ID for migrated data sets to MIGRIX:

```
MIGRATEVOL MIGRIX
```

PRIMARY Statement

Use the PRIMARY statement to specify the amount of tracks, blocks, or cylinders (according to SPACETYPE) for primary allocation.

Syntax



Parameters

amount
The amount of tracks, blocks, or cylinders. The valid range is 1 to 16777215 blocks (the operating system maximum). The default is 1.

Examples

Set the primary allocation to 5 tracks:

```
PRIMARY 5
```

Usage Notes

- If you specify no value for *amount*, FTP will not specify the amount of tracks, blocks, or cylinders for primary allocation.
- You should specify no value for *amount* if the DATACLASS statement is specified and the space allocation from the SMS data class is to be used. If the SMS data class is to be used for space allocation, both the PRIMARY and SECONDARY values must be omitted and the value on the SPACETYPE statement will be ignored.
- For allocating partitioned data sets, *amount* is the amount that will be allocated for the primary extent.
- For allocating sequential data sets, *amount* is the maximum amount that will be allocated for the primary extent. If a lesser amount is needed to hold the data being transferred, only the amount actually needed to hold the data will be allocated.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “SECONDARY Statement” on page 582
- “SPACETYPE Statement” on page 594

QUOTESOVERRIDE Statement

Use the QUOTESOVERRIDE statement to indicate the usage of single quotes appearing at the beginning of, or surrounding, a file name.

Syntax

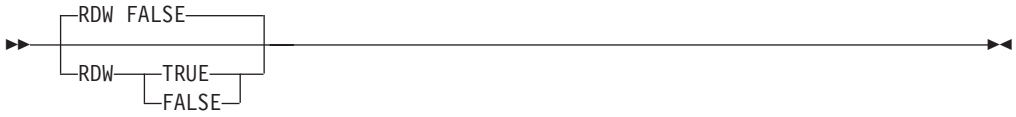
Usage Notes

- If TRUE is specified, then single quotes appearing at the beginning and end of a file name are interpreted to mean that the file name contained inside the single quotes should override the current working directory instead of being appended to the current working directory. This is the way single quotes are currently used in all previous MVS FTP servers, and is the default. Any single quotes inside the beginning and ending quote are treated as part of the file name.
- If FALSE is specified, then a single quote at the beginning of the file name, as well as all other single quotes contained in the file name, will be treated as part of the actual file name. The entire file name, including the leading single quote, will be appended to the current working directory.

RDW Statement

Use the RDW statement to specify if the RDW from variable format data sets should be retained as data.

Syntax



Usage Notes

- If TRUE is specified, RDWs are transferred as part of the data.
- If FALSE is specified, RDWs are discarded when transferring variable format data sets.

RECFM Statement

Use the RECFM statement to specify the record format of a data set.

Syntax

Parameters

format
The record format of a data set. Valid record formats are: F, FM, FA, FS, FSA, FSM, FB, FBM, FBA, FBS, FBSM, FBSA, V, VM, VA, VS, VSM, VSA, VB, VBM, VBA, VBS, VBSA, VBSM, U, UA, and UM. The default record format is VB. The meanings of the record formats are:

Format	Description
A	Records contain ISO/ANSI control
B	Blocked records
F	Fixed record length
M	Records contain machine code control characters
S	Spanned records (if variable), or Standard (if fixed)
U	Undefined record length
V	Variable record length characters

Examples

Use fixed blocked record format:

```
RECFM FB
```

Specify RECFM with no value to allow the RECFM value of a DCB data set or an SMS dataclass to be used:

```
RECFM
```

Usage Notes

- If you specify no value for *format*, no record format will be specified when allocating new data sets.
- You should specify no value for *format* if you specify the DATACLASS statement and the record format from the SMS data class is to be used.

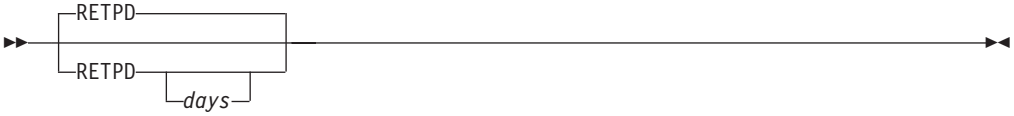
Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “DCBDSN Statement” on page 563

RETPD Statement

Use the RETPD statement to specify the number of days that a newly allocated data set should be retained.

Syntax



Parameters

days
The number of days that a newly allocated data set should be retained. The valid range is 0 through 9999. The default is to have no retention period assigned to the data set.

Examples

- Make the new data set expiration date to be 30 days:

  ```
  RETPD 30
  ```

- Use a retention period of 0 days:

  ```
  RETPD 0
  ```

Usage Notes

- If you do not specify the RETPD statement or if you specify the RETPD statement with no value, no retention period is assigned to newly allocated data sets.
- You should specify no value for *days* if the DATACLASS statement is specified and the retention period from the SMS data class is to be used.
- If you specify 0 for *days*, newly allocated data sets will be assigned a retention period of 0 days. This means that the retention period of the data set will expire on the same day that the data set is created.
- If the SMS data class or DCBDSN model data set have a retention period, this retention period can be overridden to a new retention period. The retention period cannot be overridden to have no assigned retention period.

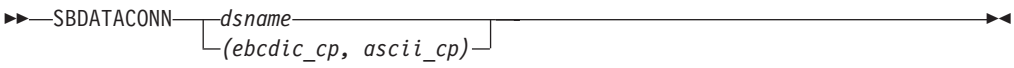
Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “DCBDSN Statement” on page 563

SBDATACONN Statement

This statement defines the conversions between EBCDIC and ASCII code pages to be used for data transfer.

Syntax



Parameters

dsname
The fully-qualified name of an MVS data set or HFS file that contains the EBCDIC to ASCII translate tables and the ASCII to EBCDIC translate tables generated by the CONVXLAT utility. For more information on translation tables, see “Chapter 36. Using Translation Tables” on page 1133.

ebcdic_cp
The name of an EBCDIC code page recognized by iconv

ascii_cp
the name of an ASCII code page recognized by iconv

Examples

```
SBDATACONN (IBM-037, IBM-850)
```

Usage Notes

- The SYSFTSX DD statement, if present, overrides the SBDATACONN statement.
- If both the SYSFTSX DD statement and the SBDATACONN statement are not present, the search order for a TCPXLBIN data set is followed. See “FTP Code Page Conversion” on page 613 for this search order. If no TCPXLBIN data set is found, the same conversion established for the control connection is used for single-byte data transfer.
- The dsname must *not* be enclosed in quotes. If quotes appear, they will be treated as part of the name.
- The HFS name is case-sensitive. The MVS name may be entered in any case.
- The length of an HFS name is limited by the record length of the FTP.DATA file because the SBDataconn statement must fit on one line.
- The HFS name cannot start with a left paren (() or contain semicolons (;) or any blanks ( ).
- If you specify SBDATACONN (*ebcdic_cp*, *ascii_cp*), FTP uses the iconv() application programming interface to do translation from EBCDIC to ASCII and ASCII to EBCDIC. The values that you enter on the SBDATACONN statement are used by FTP as parameters to the iconv() interface.

Related Topics

For the code pages supported by iconv, refer to *OS/390 C/C++ Programming Guide*.

SECONDARY Statement

Use the SECONDARY statement to specify the amount of tracks, blocks, or cylinders (according to SPACETYPE) for secondary allocation.

Syntax

Parameters

amount
The amount of tracks, blocks, or cylinders. The valid range is 0 to 16777215 blocks (the operating system maximum). The default is 1.

Examples

Set the secondary allocation to two tracks:

```
SECONDARY 2
```

Usage Notes

- If you specify no value for *amount*, FTP will not specify the amount of tracks, blocks, or cylinders for secondary allocation.
- You should specify no value for *amount* if the DATACLASS statement is specified and the space allocation from the SMS data class is to be used. If the SMS data class is to be used for space allocation, both the PRIMARY and SECONDARY values must be omitted and the value on the SPACETYPE statement will be ignored.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “DATACLASS Statement” on page 559
- “PRIMARY Statement” on page 576
- “SPACETYPE Statement” on page 594

SMF Statement

Use the SMF statement to specify the default SMF record subtype to be used for all SMF records.

Syntax

Parameters

STD
Indicates that standard SMF subtypes will be assigned for the record subtypes as follows:

- APPEND - 70
- DELETE - 71
- Login failure - 72
- RENAME - 73
- RETRIEVE - 74
- STORE - 75
- STORE UNIQUE - 75

number
The SMF record subtype to be used for all FTP server records, unless otherwise specified for a particular record subtype. The valid range is 1 through 255. There is no default value.

Examples

To have all FTP server records created with standard subtypes:

```
SMF STD
```

Usage Notes

- If the SMF statement is omitted, then SMF recording will occur for only the events that have an explicit statement coded. For example, if SMF is omitted but an SMFAPPE statement is coded, then only the APPEND command will have SMF recording.
- If the SMF statement is coded with a value of STD, then all other SMF-related statements that have a value coded (even if it is STD) are flagged with warning message EZYFT58 and their specifications are ignored. SMF STD means standard values and no other values are allowed.
- If none of the SMF subtype statements are coded in the *hlq*.FTP.DATA data set, then no SMF records are written by the FTP server.

Related Topics

- “SMFAPPE Statement” on page 585
- “SMFDEL Statement” on page 586
- “SMFJES Statement” on page 588
- “SMFREN Statement” on page 590
- “SMFRETR Statement” on page 591

- “SMFSQL Statement” on page 592
- “SMFSTOR Statement” on page 593

SMFAPPE Statement

Use the SMFAPPE statement to specify the SMF record subtype to be used for the APPE (APPEND) subcommand.

Syntax

Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for APPEND, which is a value of 70, is assigned for APPEND records.

Examples

Set the SMF record subtype for APPEND to 70:

```
SMFAPPE 70
```

Usage Notes

If you do not specify the SMFAPPE statement, the *number* on the SMF statement will be used for APPEND records if the SMF statement is specified. If neither the SMF or the SMFAPPE statement is specified, no SMF records will be collected for the APPEND subcommand.

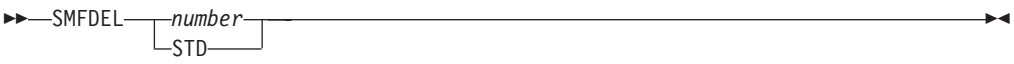
Related Topics

See “SMF Statement” on page 583.

SMFDEL Statement

Use the SMFDEL statement to specify the SMF record subtype to be used for the DELE (DELETE) subcommand.

Syntax



Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for DELETE, which is a value of 71, is assigned for DELETE records.

Examples

Set the SMF record subtype for DELETE to 71:

```
SMFDEL 71
```

Usage Notes

If you do not specify the SMFDEL statement, then the *number* on the SMF statement will be used for DELETE records if the SMF statement is specified. If neither the SMF or the SMFDEL statement is specified, no SMF records will be collected for the DELETE subcommand.

Related Topics

See “SMF Statement” on page 583.

SMFEXIT Statement

Use the SMFEXIT statement to specify that the user exit routine FTPSMFEX is called before passing the SMF record to SMF.

Syntax

Parameters

None.

SMFJES Statement

Use the SMFJES statement to specify that SMF records are collected when FILETYPE is JES (remote job submission).

Syntax

Parameters

None.

SMFLOGN Statement

Use the SMFLOGN statement to specify the SMF record subtype to be used when recording logon failures.

Syntax

Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for logon failures, which is a value of 72, is assigned for logon failure records.

Examples

Set the SMF record subtype for login failure records to 72:

```
SMFLOGN 72
```

Usage Notes

If you do not specify the SMFLOGN statement, then the *number* on the SMF statement will be used for login failure records if the SMF statement is specified. If neither the SMF or the SMLOGN statement is specified, no SMF records will be collected for login failures.

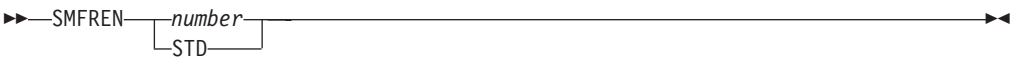
Related Topics

See “SMF Statement” on page 583.

SMFREN Statement

Use the SMFREN statement to specify the SMF record subtype to be used for the RNFT/RNTO (RENAME) subcommand.

Syntax



Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for RENAME, which is a value of 73, is assigned for RENAME records.

Examples

Set the SMF record subtype for RENAME records to 73:

```
SMFREN 73
```

Usage Notes

If you do not specify the SMFREN statement, then the *number* on the SMF statement will be used for RENAME records if the SMF statement is specified. If neither the SMF or the SMFREN statement is specified, no SMF records will be collected for the RENAME subcommand.

Related Topics

See “SMF Statement” on page 583.

SMFRETR Statement

Use the SMFRETR statement to specify the SMF record subtype to be used for the RETR (RETRIEVE) subcommand.

Syntax

Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for RETRIEVE, which is a value of 74, is assigned for RETRIEVE records.

Examples

Set the SMF record subtype for RETRIEVE records to 74:

```
SMFRETR 74
```

Usage Notes

If you do not specify the SMFRETR statement, then the *number* on the SMF statement will be used for RETRIEVE records if the SMF statement is specified. If neither the SMF or the SMFRETR statement is specified, no SMF records will be collected for the RETRIEVE subcommand.

Related Topics

See “SMF Statement” on page 583.

SMFSQL Statement

Use the SMFSQL statement to specify that SMF records are collected when FILETYPE is SQL (SQL query function).

Syntax

Parameters

None.

SMFSTOR Statement

Use the SMFSTOR statement to specify the SMF record subtype to be used for the STOR (STORE) and STOU (STORE UNIQUE) subcommands.

Syntax

Parameters

number
The SMF record subtype. The valid range is 1 through 255. There is no default value, however, if the SMF statement is coded, the value specified for the SMF statement will be used as the default.

STD
Indicates that standard SMF subtype for STORE and STORE UNIQUE, which is a value of 75, is assigned for STORE and STORE UNIQUE records.

Examples

Set the SMF record subtype for STORE and STORE UNIQUE records to 75:

```
SMFSTOR 75
```

Usage Notes

If you do not specify the SMFSTOR statement, then the *number* on the SMF statement will be used for STORE and STORE UNIQUE records if the SMF statement is specified. If neither the SMF or the SMFSTOR statement is specified, no SMF records will be collected for the STORE subcommand.

Related Topics

See “SMF Statement” on page 583.

SPACETYPE Statement

Use the SPACETYPE statement to specify whether newly allocated data sets are allocated in blocks, cylinders, or tracks.

Syntax

Parameters

BLOCK
Use blocks when allocating new data sets.

CYLINDER
Use cylinders when allocating new data sets.

TRACK
Use tracks when allocating new data sets. This is the default.

Examples

Allocate data sets in tracks:

```
SPACETYPE TRACK
```

Usage Notes

If you do not give values on the PRIMARY and SECONDARY statements in order to use the SMS data class, the value on the SPACETYPE statement will be ignored and SMS will determine the spacetype.

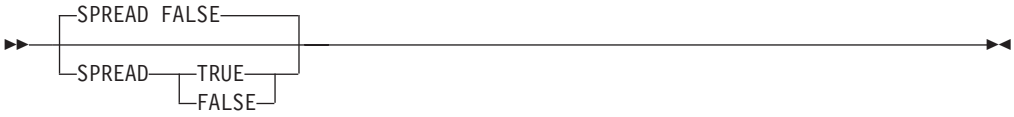
Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “PRIMARY Statement” on page 576
- “SECONDARY Statement” on page 582

SPREAD Statement

Use the SPREAD statement to specify whether or not the output is in spreadsheet format when the file type is SQL.

Syntax



Parameters

TRUE
Specifies that the output is in spreadsheet format.

FALSE
Specifies that the output is not in spreadsheet format. This is the default.

Examples

Make the output be in spreadsheet format:

```
SPREAD TRUE
```

SQLCOL Statement

Use the SQLCOL statement to specify the column headings of the output file.

Syntax

Parameters

ANY
Use the label, but if there is no label, the name becomes the column heading.

LABELS
Use the label of the column headings. If any of the columns do not have labels, the server uses COL*number*, where *number* is the column number reading left to right.

NAMES
Use the name of the column headings and ignore the labels. This is the default.

Examples

Use the label of the column headings:

```
SQLCOL LABELS
```

STARTDIRECTORY Statement

Use the STARTDIRECTORY statement to specify which file system will be initially used when a new user logs in.

Syntax

Parameters

HFS
Use the Unix System Services hierarchical file system (HFS). The initial directory will be the user's root directory in the HFS.

MVS
Use MVS partitioned data sets. The initial data set name will have a prefix of the userid.

Examples

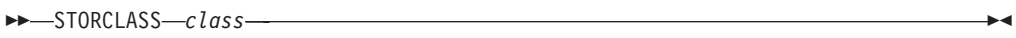
Set the initial user directory to the user's root directory in the HFS:

```
STARTDIRECTORY HFS
```

STORCLASS Statement

Use the STORCLASS statement to specify the SMS storage class as defined by your organization for the FTP server.

Syntax



Parameters

class
The SMS class.

Examples

Use the SMS storage class SMSSTOR when allocating new data sets:

```
STORCLASS SMSSTOR
```

Related Topics

See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.

TRACE Statement

Use the TRACE statement to start tracing for the FTP server. The trace output will be written to syslog.

Syntax

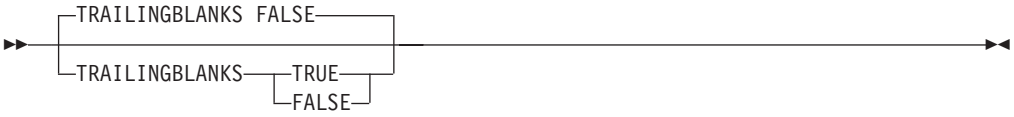
Parameters

None.

TRAILINGBLANKS Statement

Use the TRAILINGBLANKS statement to specify whether trailing blanks in a fixed format data set are transferred when the data set is transferred.

Syntax



Parameters

TRUE
Specifies that the trailing blanks in a fixed format data set are included when the data set is retrieved.

FALSE
Specifies that the trailing blanks in a fixed format data set are not retrieved. This is the default.

Usage Notes

Retrieve the fixed format data set and include trailing blanks:

```
TRAILINGBLANKS TRUE
```

UCSHOSTCS Statement

Use the UCSHOSTCS statement to specify the EBCDIC code set to be used for data conversion to or from Unicode. If the UCSHOSTCS statement is not used, the current code set for the FTP server host will be used.

Syntax

Parameters

code_set
The EBCDIC code set that is to be used when converting to or from Unicode. (See ″Code Set Converters Supplied″ in the *OS/390 C/C++ Programming Guide* for the valid EBCDIC code set names.)

UCSSUB Statement

Use the UCSSUB statement to specify whether Unicode-to-EBCDIC conversion should use the EBCDIC substitution character or cause the data transfer to be terminated if a Unicode character cannot be converted to a character in the target EBCDIC code set.

Syntax

Parameters

TRUE
Specifies that the EBCDIC substitution character will be used to replace any Unicode character that cannot successfully be converted. Data transfer will continue.

FALSE
Specifies that the data transfer will be terminated if any Unicode character cannot be successfully converted.

UCSTRUNC Statement

Use the UCSTRUNC statement to specify whether the transfer of Unicode data should be aborted if truncation occurs at the MVS host. (Truncation can occur if the lrecl of the receiving data set is not large enough to contain a line of Unicode data after it has been converted to EBCDIC.)

UCSTRUNC applies to inbound data transfers only.

Syntax

Parameters

TRUE
Specifies that truncation is allowed. The data transfer will continue even if EBCDIC data is truncated.

FALSE
Specifies that truncation is not allowed. The transfer is to be aborted if the lrecl of the receiving data set is too small to contain the data after conversion to EBCDIC.

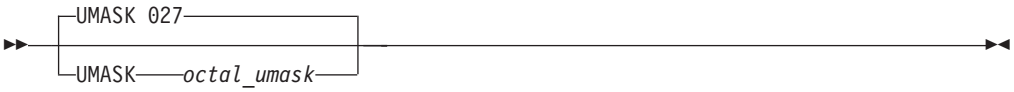
Note: The setting of CONDDISP will determine what happens to the target data set if the transfer is aborted.

UMASK Statement

Use the UMASK statement to define the file mode creation mask.

The file mode creation mask defines which permission bits are NOT to be set on when a file is created. When a file is created, the permission bits requested by the file creation are compared to the file mode creation mask, and any bits requested by the file creation which are disallowed by the file mode creation mask are turned off.

Syntax



Parameters

octal_umask
is the octal umask.

Examples

When a file is created the permission bits for file creation are 666 (-rw-rw-rw-). If the file mode creation mask is 027, the requested permissions and the file mode creation mask are compared:

```
110110110   - 666
000010111   - 027
---------
110100000   - 640
```

When the UMASK is set to 027, the actual permission bits set for a file when it is created will be 640 (-rw-r-----).

Usage Notes

You cannot use FTP to create HFS files having execute permissions. If you require execute permissions, use the *site chmod* command after the file is created. For more information on *site chmod*, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

UNITNAME Statement

Use the UNITNAME statement to specify the unit type for allocation of new data sets.

Syntax

Parameters

type
The type of either direct access or tape devices.

SYSDA
If *type* is not specified, SYSDA is the default.

Examples

- Set the unit type for new data sets to 3380:

```
UNITNAME 3380
```

- Set the unit type for new data sets to TAPE:

```
UNITNAME TAPE
```

Usage Notes

- If you do not use the UNITNAME statement to specify the *type*, then the unit type used for allocation is the system default unit.
- If the STORCLASS statement is also specified, the SMS storage class might contain settings that will override the UNITNAME *type*.
- It is preferable that you do not use the UNITNAME statement if you are using an SMS storage class.
- The UNITNAME can name a dynamic device.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “STORCLASS Statement” on page 598

VOLUME Statement

Use the VOLUME statement to specify the volume serial number for allocation of new data sets.

Syntax

Parameters

name
The volume serial number. The value specified for *name* is case-sensitive.

Examples

Set the volume name for new data set to WRKLB4:

```
VOLUME WRKLB4
```

Usage Notes

- If you do not use the VOLUME statement to specify the *name*, then the volume serial number used for allocation is the system default volume list.
- If the STORCLASS statement is also specified, the SMS storage class might contain settings that will override the VOLUME *name*.
- It is preferable that you do not use the VOLUME statement if you are using an SMS storage class.
- When transferring a variable-length file to multiple volumes on MVS, only the last file will contain the correct DCB characteristics.

Related Topics

- See “Storage Management Subsystem (SMS)” on page 533 for more information about specifying attributes when allocating new data sets.
- “STORCLASS Statement” on page 598

WLMCLUSTERNAME Statement

Use the WLMCLUSTERNAME statement to instruct the FTP Daemon to register in a DNS/WLM sysplex connection balancing group. This would be necessary if FTP is to participate in a group of FTP servers in the same sysplex which will allow clients to attach to them in a connection balanced manner. See “Chapter 21. Configuring the Bind-Based Domain Name System (DNS)” on page 709 for further discussion of this topic. This statement may be repeated up to 16 times to cause the FTP daemon to register in up to 16 different groups.

Syntax

Parameters

ftp_group_name
Any legal DNS name up to 18 characters. All servers in a group must register with this name.

Examples

Allow an FTP client to connect to any of a number of equivalent FTP servers in an MVS sysplex by using the name ftpgroup instead of the name or IP address of a particular host in that sysplex.

```
WLMCLUSTERNAME ftpgroup
```

WRAPRECORD Statement

Use the WRAPRECORD statement to specify whether data will be wrapped to the next record or truncated if no new line character is encountered before the logical record length is reached.

Syntax

Parameters

TRUE
Indicates that data will be wrapped to the next record if no new line character is encountered before the logical record length is reached.

FALSE
Indicates that data will be truncated if no new line character is encountered before the logical record length is reached. This is the default.

Examples

Truncate data if no new line character is encountered before the logical record length is reached:

```
WRAPRECORD FALSE
```

XLATE Statement

Use the XLATE statement to specify a data set containing translate tables to be used for the data connection.

Syntax

Parameters

name
Specifies a 1-8 character name that corresponds to a data set that contains translate tables.

FTP looks first for an environment variable called _FTPXLATE_*name*. If the environment variable exists, its value is used as the data set name.

Note: The environment variable name must be all uppercase, although the XLATE parameter can be in mixed case.

If the environment variable does not exist FTP looks for a data set called *hlq*.*name*.TCPXLBIN.

Examples

```
XLATE FRED
```

If environment variable _FTPXLATE_FRED=FREDDYS.TABLES is defined for the FTP server, the statement above specifies that the translate tables in data set FREDDYS.TABLES should be used for the data connection.

If there is no such environment variable defined, the above statement specifies that the translate tables data set *hlq*.FRED.TCPXLBIN should be used.

Usage Notes

- CCXLATE and CTRLCONN are mutually exclusive statements. If both statements appear in your FTP.DATA file, XLATE will be ignored.
- The XLATE statement (and its value) is not case-sensitive, but the name of the corresponding environment variable must be all uppercase or FTP will not recognize it.

Related Topics

- “Defining Optional Environment Variables for the FTP Server” on page 527
- “CCXLATE Statement” on page 555
- To see the search order that determines the conversion for the control connection, see “FTP Code Page Conversion” on page 613.

Starting, Stopping, and Tracing the FTP Server

The FTP server can be started as a started task or from the OS/390 shell. It can also be started automatically when OS/390 UNIX or TCP/IP is started.

The FTP server offers MODIFY command support to start and stop trace dynamically. Use of this support is restricted to the users with MODIFY command privilege.

Starting the FTP Server

The FTP server uses the ETC.SERVICES file to determine which port address to use for the FTP control port. If desired, the control port for the FTP server may also be specified using the PORT start option of the FTP server. If the PORT start option is specified when starting the FTP server, this value will override the value specified in the ETC.SERVICES file. If there is no entry for the FTP server in the ETC.SERVICES file, and the PORT start option is not specified, the default ports for the FTP server are port 21 for the control port and port 20 for the data port.

During initialization, the jobname of the server changes from the original jobname to a new jobname:

- The ***original*** jobname is used as the hlq for the initialization and config data sets.
- The ***new*** job name is used for operator STOP and MODIFY commands.

The new job name and process ID are logged in syslogd in message EZYFT41I.

Starting FTP as a Started Task

Use the START command to start the FTP server as a started task:

START command syntax

where:

procname
: The name of the FTP cataloged procedure library. See “FTP Server Cataloged Procedure (FTPD)” on page 524 and “Specifying the FTPD Parameters” on page 525.

Starting FTP from the OS/390 Shell

From the OS/390 shell, invoke the ftpd module. For example, to start an instance of FTP using port 8097, use the following:

```
/usr/sbin/ftpd   port 8097
```

Note: You must be in superuser when you start the FTP daemon from the shell, or the server will not be able to verify passwords.

Starting FTP Automatically

To start FTP automatically, either:

- Autolog FTP when TCP/IP is started (See “Step 1: Specify AUTOLOG, PORT, and KEEPALIVEOPTIONS Information” on page 523).
- Start FTP when OS/390 UNIX is started by adding the following to /etc/rc:

```
export _BPX_JOBNAME='FTPD'
/usr/sbin/ftpd <start parameters> &
```

Notes:

- Syslogd should be started before FTP, or all messages and trace entries will appear on the MVS system console.
- If TCP/IP initialization is not complete before FTP is started, the FTP server will be unable to establish a socket and the following message will be sent to syslogd and written to the appropriate HFS file:

  ```
  EZYFT12E socket error: EDC5112I Resource temporarily unavailable
  ```

  The FTP server will continue to try every minute until TCP/IP initialization is complete, at which time FTP initialization can complete. FTP will not recognize a stop command at this stage of its initialization, but you can issue a cancel command.

FTP Server Exit Codes

FTP uses the following error exit codes:

Exit Code	Explanation
12	Daemon initialization failed, or unable to accept an incoming connection. An EZY message identifying the specific problem is issued to syslog.
24	Client session initialization terminated because the FTP server load module cannot be loaded or executed. Message EZYFT53E is issued to syslog.
28	The FTP server is not available because IBM’s TCP/IP is not enabled.

Stopping the FTP Server

The main FTP server process can be stopped either by issuing the MVS STOP command to the new jobname or by issuing an OS/390 shell ″kill″ command for the process ID. Stopping the main FTP server process will not affect the active session processes. The active session processes will remain active until the client/server session is terminated by an FTP QUIT command or until the client process is stopped.

The individual client session processes cannot be stopped with an MVS STOP command. The client process can only be stopped by issuing an OS/390 shell ″kill″ command for the client session process. To aid in killing the client session processes, the process ID for the client will be returned in the STAT command output.

Tracing the FTP Server

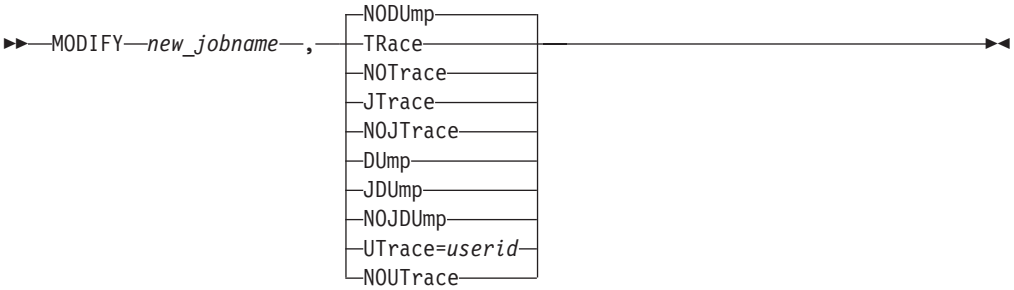
Use the TRACE start parameter, or the TRACE statement in FTP.DATA, to start tracing during FTP initialization

Use the MODIFY command to start and stop tracing after initialization is complete:

Only FTP sessions established after trace is active can be traced. When tracing is stopped, sessions currently connected to the server will continue to be traced. New FTP sessions will not be traced.

Syntax

MODIFY command syntax



Parameters

TRACE
Enables a general trace for all clients connecting to the server.

NOTRACE
Disables the general trace.

JTRACE
Enables a JES trace for JES-related activity for all clients connecting to the server.

NOJTRACE
Disables the JES trace.

DUMP
Includes additional detailed activity in the log whenever the general trace is active.

NODUMP
Excludes detailed data from the general trace log. This is the default.

JDUMP
Includes additional detailed activity in the log whenever the JES trace is active.

NOJDUMP
Excludes detailed data from the JES trace log.

UTRACE=*userid*
Enables all levels of tracing for one specified user ID. All other trace options that were in effect are suspended.

NOUTRACE
Disables tracing for the specified user ID and resumes other traces that were in effect.

Examples

Enable JES tracing for an FTP server started with the FTPD procedure:

```
F FTPD1,JTRACE
```

where FTPD1 is the new jobname as given in EZYFT41I.

Related Topics

For more information on FTP server traces and how the parameter values interact, refer to *OS/390 SecureWay Communications Server: IP Diagnosis*.

FTP Code Page Conversion

Code page conversion must be performed between EBCDIC and ASCII for:

- FTP subcommands and replies sent over the control connection
- Data transferred over the data connection

FTP uses *iconv* functions to establish ASCII-to-EBCDIC and EBCDIC-to-ASCII translate tables for the control connection and the data connection, with a default of 7-bit ASCII. In addition, FTP maintains support for the use of translate tables generated by the CONVXLAT utility for the control or data connection.

Note: Using iconv conversion to retrieve EBCDIC data that was created with CONVXLAT-generated conversion tables could result in data corruption due to possible conversion table differences.

Once an end user has logged in, a 'site' subcommand can be used to change the ASCII being used on the control connection or the single byte translation for the data connection.

Code Page Conversions for the Control Connection

For the EBCDIC/ASCII conversions for the control connection, FTP uses either iconv or the existing support for single-byte translation tables.

Priority

The priority for establishing the EBCDIC/ASCII control connection is:

1. CTRLCONN or CCXLATE keyword in FTP.DATA
2. Search order used to locate a TCPXLBIN data set:
   a. original jobname.SRVRFTP.TCPXLBIN
   b. hlq.SRVRFTP.TCPXLBIN
   c. original jobname.STANDARD.TCPXLBIN
   d. hlq.STANDARD.TCPXLBIN
3. 7-bit ASCII
4. Internal (hard-coded) 7-bit tables

Code Page Conversions for the Data Connection

For the transfer of data on the data connection, FTP will support:

1. All single-byte conversions available through *iconv*. For example, Country-Extended_code-Pages (CECPs) <-> ISO8859-1 and IBM-1047 <-> IBM-850 conversions are available for data transfers.

   Note: The double-byte iconv convertors are not supported by FTP.
2. Both single-byte and double-byte data conversions are supported with the translate tables provided with TCP/IP or generated by the CONVXLAT utility.

Priority

The priority for establishing ASCII/EBCDIC conversions for the data connection is:

1. SYSFTSX DD statement in the startup proc, where the named data set contains CONVXLAT-generated translate tables. The data set can be an MVS data set or an HFS file.
2. SBDATACONN or XLATE keyword in FTP.DATA.
3. Search order to locate a TCPXLBIN data set, where the MVS data set contains CONVXLAT-generated translate tables:
   a. original jobname.SRVRFTP.TCPXLBIN
   b. hlq.SRVRFTP.TCPXLBIN
   c. original jobname.STANDARD.TCPXLBIN
   d. hlq.STANDARD.TCPXLBIN
4. The same conversions established for the control connection.

Chapter 15. Configuring the Trivial File Transfer Protocol Server

TFTP is a TCP/IP protocol that is used to transfer files. TFTP can read or write files from or to a remote server. On the S/390 system, TFTP is a server that you can configure with the command line option during TFTP invocation.

Considerations for OS/390

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory.

CAUTION:
The TFTP server uses well-known port 69. The TFTP server has no user authentication. Any client that can connect to port 69 on the server has access to TFTP. If the TFTP server is started without a directory, it allows access to the entire HFS. To restrict access to the HFS, start the TFTP server with a list of directories.

To start the TFTP server from the command line, type the tftpd command.

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit]
   [-s maxsegsize] [-f file] [-a archive directory [-a ...]]
   [directory ...]
```

Following are the parameters used for the tftpd command:

-l Logs all the incoming read and write requests and associated information to the system log. Logged information includes the IP address of the requestor, the file requested, whether the request was successful.

-p port
Uses the specified port. The TFTP server usually receives requests on well-known port 69. You can specify the port in which requests are to be received.

-t timeout
Sets the packet timeout. The TFTP server usually waits 5 seconds before presuming that a transmitted packet has been lost. You can specify a different timeout period in seconds.

-r maxretries
Sets the retry limit. The TFTP server usually limits the number of retransmissions it performs because of lost packet to 5. You can specify a different retry limit.

-c concurrency_limit
Sets the concurrency limit. The TFTP server spawns both threads and processes to handle incoming requests. You can specify the limit for the number of threads that may be concurrently processing requests under a single process. When the limit is exceeded, a new process is spawned to handle requests. The default is 200 threads.

-s maxsegsize
Sets the maximum block size that can be negotiated by the TFTP block size option. The default is 8192.

-f file Specifies a cache file. You can specify a file containing information on files

to be pre-loaded and cached for transmission. A cache file consists of one or more entries. For clarity, place each entry on a separate line. An entry has the form:

a | b <pathname>

where:

- *a* indicates that the specified file is cached in ASCII form. The file is preconverted to netascii format.
- *b* indicates that the specified file is cached in binary form, with no conversion.

Following are examples of cache file entries,

```
a  /usr/local/textfile
b  local/binaryfile
```

If a relative pathname to the file is specified, the TFTP server searches the specified directories for the file.

The cached version of a file is only used for requests requiring the specified format. For example, the binary cached version of a file is not used in satisfying a request for the file in netascii format. If a file is to be retrieved in both binary and ASCII formats, the user must specify that two copies of the file be cached with one in binary format, and the other in netascii format.

Caching is not dynamic. The cache files are read in when the TFTP server is started and are not updated, even if the file on disk is updated. To update or refresh the cache, the TFTP server must be recycled.

-a archive directory

Specifies an archive directory. The files in this directory and its subdirectories are treated as binary files for downloading. This option is useful on EBCDIC machines that act as file servers for ASCII clients. Multiple -a options can be specified; one directory per -a option. Directories must be specified as absolute pathnames. You can specify no more than 20 directories.

directory

Specifies an absolute path name for a directory. You may specify no more than 20 directories on the tftpd command line.

If the TFTP server is started without a list of directories, all mounted directories are considered active.

If a list of directories is specified, only those specified directories are active. That list is used as a search path for incoming requests that specify a relative path name for a file.

Activating a directory activates all of its subdirectories.

For a file to be readable by the TFTP server, the file must be in an active directory and have world (″other″) read access enabled. For a file to be writable by the TFTP server, the file must already exist in an active directory and have world (″other″) write access.

The TFTP server pre-forks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist.

In case of a flood of concurrent TFTP requests, the TFTP server may fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

To terminate the TFTP server, send a SIGTERM signal to the oldest existing TFTP process. This is the process that has a parent process ID of 1. Termination of this process will cause all of its children to terminate.

Chapter 16. Configuring the TIMED Daemon

TIMED is a TCP/IP daemon that is used to provide the time. TIMED gives the time in seconds since midnight January 1, 1900. You can start TIMED from the OS/390 shell or as a started procedure. Each of these methods is described below.

Starting TIMED from OS/390

TIMED is installed in the /usr/lpp/tcpip/sbin/ directory.

To start the TIMED server from the command line, type the timed command.

```
timed [-l] [-p port]
```

Following are the parameters used for the timed command:

-l Logs all the incoming requests and responses to the system log. Logged information includes the IP address of the requestor.

-p port
Uses the specified port. The TIMED server usually receives requests on well-known port 37. You can specify the port in which requests are to be received.

Starting TIMED as a Procedure

The following sample shows how to start TIMED as a procedure.

```
//TIMED    PROC
//*
//* TCP/IP for MVS Network Station Manager Feature
//* SMP/E distribution name: EZATTMDP
//*
//*   5645-001 5655-HAL (C) Copyright IBM Corp. 1997.
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//* Function: Time server start procedure
//*
//TIMED    EXEC PGM=TIMED,REGION=0K,TIME=NOLIMIT,
//         PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/'
//*STEPLIB DD DISP=SHR,DSN=TCP.SEZALINK,
//*           VOL=SER=,UNIT=
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//         PEND
```

Chapter 17. Configuring the OS/390 Unix Remote Execution Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure and operate the OS/390 UNIX Remote Execution server.

The Remote Execution Protocol Daemon (REXECD) is the server for the REXEC routine. REXECD allows execution of OS/390 UNIX commands with authentication based on user names and passwords.

The Remote Shell Server (RSHD) is the server for the remote shell (RSH) client. The server provides remote execution facilities with authentication based on privileged port numbers, user IDs, and passwords.

Installation Information

This section describes the HFS files used by OS/390 UNIX REXECD and RSHD.

HFS Files for OS/390 UNIX REXECD

Note: The userid associated with the daemon in /etc/inetd.conf requires superuser authority. See ″Setting Up for Daemons″ in *OS/390 UNIX System Services Planning* for a description of the kinds of authority defined for daemons.

The HFS files used by OS/390 UNIX REXECD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orexecd
The server.

If BPX.DAEMON is specified, then the sticky bit must be set on, and /usr/sbin/orexecd and orexecd must reside in an authorized MVS data set.

/usr/lib/nls/msg/C/rexdmsg.cat
The message catalog used by the OS/390 UNIX REXECD server.

Where the server looks for the message catalog (rexdmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If rexdmsg.cat does not exist, the software

will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

HFS Files for OS/390 UNIX RSHD

The HFS files used by OS/390 UNIX RSHD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orshd
The server.

If BPX.DAEMON is specified, the sticky bit must be set on, and /usr/sbin/orshd and orshd must reside in an authorized MVS data set.

/usr/sbin/ruserok
An optional user exit that will authenticate users logging into the OS/390 UNIX RSHD server with a null password. See "Setting up the OS/390 UNIX RSHD Installation Exit" below for more information.

Note: This exit is required to allow support for null passwords with RSH.

/usr/lib/nls/msg/C/rshdmsg.cat
The message catalog associated with the OS/390 UNIX RSHD client is stored here. If this file does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

Setting up the OS/390 UNIX RSHD Installation Exit

When the -r option is enabled, if there is no password specified on the RSH command from the client, OS/390 UNIX RSHD will drive the installation exit. When the installation exit is driven, RSHD looks for a program in /usr/sbin named ruserok. This is the only name that it will look for. If /usr/sbin/ruserok is not found, the request will fail.

When the OS/390 UNIX RSHD server invokes /user/sbin/ruserok, it will pass parameters in the following order:

1. hostname
2. local user's UID
3. remote userid
4. local userid

If OS/390 UNIX RSHD receives a return code of zero from the installation exit, OS/390 UNIX RSHD continues. Any non-zero return code from the installation exit will cause RSHD to issue message EZYRS25E to the client and terminate all

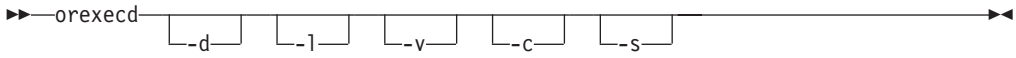
connections. The following code fragment can be used as an example to begin building a working ruserok installation exit:

```
int main(argc, argv)
     int argc;
     char *argv[];
  char *rhost1;    /* "hostname" or "hostname.domain" of client
                      obtained by caller:
                      gethostbyaddr(getpeername()) */
  int locuid;      /* uid of the user name on local system */
  char *cliuname;  /* user name on client's system */
  char *servuname; /* user name on this (server's) system */
  int rc = 4;

  rhost1 = argv[1];
  locuid = atoi(argv[2]);
  cliuname = argv[3];
  servuname = argv[4];
  .
  <authenticate user and set rc=0 if valid>
  .
  return(rc);
```

OS/390 UNIX REXECD Command (orexecd)

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke orexecd.

Following is the syntax for the orexecd command:



Following is a description of the supported options:

Option Description

-d Print debug information to syslogd.

-l Write each successful login to syslogd with the remote user, remote system, local user, and the command executed.

-v Write the title and ptf level to syslogd.

-c Write all messages in uppercase.

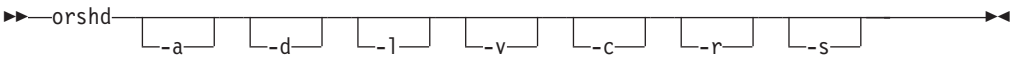
-s Invoke the remote shell as a login shell (that is, run /etc/profile and $HOME/.profile).

OS/390 UNIX RSHD Command (orshd)

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke otelnetd.

Note: The userid associated with the daemon in /etc/inetd.conf requires superuser authority. See ″Setting Up for Daemons″ in *OS/390 UNIX System Services Planning* for a description of the kinds of authority defined for daemons.

Following is the syntax for the orshd command:



Following is a description of the supported options:

Option Description

-a Look up hostname and check that the address and hostname correspond.

-d Print debug information to syslogd.

-l Write each successful login to syslogd with the remote user, remote system, local user, and the command executed.

-v Write the title and ptf level to syslogd.

-c Write all messages in uppercase.

-r If a client passes a null password, invoke the /usr/sbin/ruserok user exit to authenticate the userid.

-s Invoke the remote shell as a login shell (that is, run /etc/profile and $HOME/.profile).

Chapter 18. Configuring the Remote Execution Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

The Remote Execution server allows execution of a TSO command that has been received at a remote host. This server runs the Remote EXecution Command Daemon (REXECD) which supports both the Remote Execution (REXEC) and Remote Shell (RSH) protocols.

This chapter describes how to configure and operate the Remote Execution server.

Configuration Process

Steps to configure the Remote Execution server:

1. Update AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set.
2. Determine whether the Remote Execution client will send a Remote Execution (REXEC) command or Remote Shell (RSH) command.
3. Permit remote users to access MVS resources. (Required only if the client is not sending a password.)
4. Update the Remote Execution cataloged procedure.
5. Create a user exit routine (optional).

Step 1: Specify the AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP

If you want the Remote Execution server to start automatically when the TCPIP address space is started, include the name of the member containing the RXSERVE catalogued procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  RXSERVE
ENDAUTOLOG
```

To ensure that port 512 is reserved for the Remote Execution protocol and port 514 for the Remote Shell protocol, add the name of the member containing the Remote Execution catalogued procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  512 TCP RXSERVE
  514 TCP RXSERVE
```

See “AUTOLOG Statement” on page 137 for more information about the AUTOLOG statement. See “PORT Statement” on page 229 for more information about the PORT statement.

Step 2: Determine Whether Remote Execution Client Will Send REXEC or RSH Commands

The Remote Execution client can send commands to the Remote Execution server by the following methods:

1. Sending the Remote Execution (REXEC) command
2. Sending the Remote Shell (RSH) command with a user ID and password separated by a slash (/) character with the -l option on the RSH command
3. Sending the Remote Shell (RSH) command without a password

With methods 1 and 2, the Remote Execution server executes the request and passes the password to MVS for verification. (REXEC commands require a password.) When these methods are used, skip Step 3.

With method 3, to enable an RSH client to send RSH commands to the MVS Remote Execution server without specifying a password, Step 3 is required.

Step 3: Permit Remote Users to Access MVS Resources

Follow Step 3 *only if* you have decided it is required from Step 2.

Use the following steps to ensure that the server can correctly access necessary MVS resources. You can use IBM Remote Access Control Facility (RACF) or an equivalent security program.

1. Verify that your system has been configured for allowing surrogate job submission as described in *RACF Security Adminstrator's Guide* (SC23-3726) or by using an equivalent security program.
2. Authorize the Remote Execution Server to submit jobs for the MVS user ID specified with the -l option of the RSH command. This can be done with the RACF facility as described in *RACF General User's Guide* (SC23-3728), or by using an equivalent security program.
3. Define an *mvs_userid*.RHOSTS.DATA data set.

   This data set identifies the Remote Execution clients that can execute MVS commands remotely by sending an RSH command.

   When a Remote Execution client sends an RSH request to the Remote Execution server, the request includes the local user ID of the client user (*local_userID*) and, if the client user specified the -l option of the RSH command, the request also contains the user ID to use on the remote host (*mvs_userid*). If the client does not specify the -l option, the user ID to be used on the remote host is assumed to be the same as the *local_userID*.

   When the Remote Execution server receives an RSH command without a password, the server looks for a data set called *mvs_userid*.RHOSTS.DATA. If the data set exists, the server reads it and looks for an entry with a host name that matches the client user's host. If the user ID specified on this entry in the RHOSTS.DATA data set matches the *local_userID* passed on the RSH command, the RSH command continues processing. If the entry does not exist, the server responds to the client with message `EZA4386E Permission denied`.

   In the following example of an RHOSTS.DATA data set, the MVS client user `mvsuser` is allowed to issue the RSH command without a password from host `rs60007` with a local AIX user ID of `mvsuser`.

   Example of mvsuser.RHOSTS.DATA data set:

   ```
   rs60007.itso.ral.ibm.com mvsuser
   ```

Step 4: Update the Remote Execution Cataloged Procedure

Update the Remote Execution cataloged procedure by copying the sample provided in *hlq*.SEZAINST(RXPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify the Remote Execution server parameters and modify the JCL as required for your installation.

Remote Execution Cataloged Procedure (RXPROC)

```
//RXSERVE PROC MODULE='RSHD',
//          EXIT=,
//          TSOPROC=IKJACCNT,
//          MSGCLASS=H,
//          TSCLASS=A,
//          MAXCONN=512,
//          TRACE=
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEB02V
//*
//*        5655-HAL (C) Copyright IBM Corp. 1991, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//*
//* Change Activity =
//* CFD List:
//* $01=PN64129 TCPV3R2 941207  JRC: Change defaults and descriptions
//*                                   for MSGCLASS and TSCLASS.
//* $02=PN73459 TCPV3R2 950831  JB:  Correct descriptions and defaults
//*                                   for MSGCLASS and TSCLASS.
//* End CFD List:
//*//* Supported PARMS (separated by commas - ',') are:
//*        EXIT=exitmod  - Name of an exit routine to alter JOB and
//*                        EXEC parameters for submission of TSO batch
//*                        jobs submitted for remote commands.
//*                        EXIT=NOEXIT can be specified when no exit
//*                        is required.
//*        TSOPROC=proc  - The name of the TSO batch procedure. The
//*                        default is IKJACCNT,  and it can be modified
//*                        in the exit routine specified with the EXIT
//*                        parameter.
//*        MSGCLASS=c    - The MSGCLASS parameter for TSO batch jobs
//*                        submitted to execute remote commands.
//*                        Specify a HELD class for this parameter.@02A
//*                        The default is H.  The parameter is not @02C
//*                        to be altered by the exit routine.      @01A
//*        TSCLASS=c     - The SYSOUT class for the SYSTSPRT DD for
//*                        submitted jobs. The default is A.
//*        MAXCONN=n     - The maximum number of open sockets at any
//*                        one time.  Usually each client requires 2
//*                        sockets while the command is being processed
//*                        and the output is being returned. The
//*                        default and minimum value is 512. If a
//*                        value of less than 512 is specified, the
//*                        default will be used.
//*        TRACE=options - The following options are supported:
//*                        LOG | NOLOG: controls writing trace records
//*                        on SYSPRINT.
//*                        SEND | NOSEND: controls sending trace
//*                        records to the client.
//*                        CLIENT=client | ALLCLIENTS: selects a client
//*                        host for which trace records are produced,
```

```
//*                        or ALLCLIENTS to trace all clients.
//*                        RESET: sets the options to NOLOG,NOSEND,
//*                        ALLCLIENTS.
//*                        If more than one option is specified,
//*                        enclose the options in parentheses.
//* These parameters can also be changed with a MODIFY command.
//*//RXSERVE  EXEC PGM=&amp;MODULE,PARM=('EXIT=&amp;EXIT,TSOPROC=&amp;TSOPROC',
//             'MSGCLASS=&amp;MSGCLASS,TSCLASS=&amp;TSCLASS',
//             'MAXCONN=&amp;MAXCONN,TRACE=&amp;TRACE'),
//             REGION=7500K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link
//*        list or add them to the STEPLIB definition here.  If you
//*        add them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*//*        SYSPRINT contains runtime diagnostics from RSHD. It can be
//*        a data set or SYSOUT.
//*//SYSPRINT DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//*      SYSTCPD explicitly identifies which data set is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the TSO logon
//*      procedure or in the JCL of any client or server executed
//*      as a background task.  The data set can be any sequential
//*      data set or a member of a partitioned data set (PDS).
//*
//*      For more information please see "Understanding TCP/IP Data
//*      Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Specify the Remote Execution Server Parameters

The system parameters required by the Remote Execution server are passed by the PARM operand of the EXEC statement in the Remote Execution cataloged procedure. Update the following parameters as required by your installation:

EXIT=
Name of a user exit routine to inspect and alter JOB and EXEC parameters prior to submission of TSO batch jobs initiated by remote commands.

TSOPROC=
The name of the TSO batch procedure. The default is IKJACCNT. The name IKJACCNT can be modified in the exit routine specified with the EXIT parameter.

MSGCLASS=
The MSGCLASS parameter for TSO batch jobs submitted to execute remote commands. The default is H, which points to a HELD output class. The parameter must not be altered by the exit routine.

TSCLASS=
The SYSOUT class for the SYSTSPRT DD statement for submitted jobs. The default is A.

MAXCONN=
The maximum number of open sockets at any one time. Usually, each client requires 2 sockets while the command is being processed and the output is being returned. The minimum acceptable value is 512. This is also the default.

TRACE=
The trace options that are to be in effect for the Remote Execution server.

Note: If more than one trace parameter is specified, enclose the parameters in parentheses.

LOG
Specifies trace records written to SYSPRINT.

NOLOG
Specifies no trace records written to SYSPRINT.

SEND
Specifies trace records sent to the client.

NOSEND
Specifies no trace records sent to the client.

CLIENT=*client*
Specifies a specific client host for which trace records are to be produced.

ALLCLIENTS
Specifies that trace records are to be produced for all clients.

RESET
Sets the trace options to NOLOG, NOSEND, ALLCLIENTS.

All but the MAXCONN parameter can be changed dynamically with the MODIFY command.

Step 5: Create a User Exit Routine (Optional)

Optionally, you can provide a user exit routine. This routine can alter the JOB and EXEC statement parameters to meet installation-specific requirements prior to submission of the TSO batch job.

The user exit should have the AMODE(31) attribute to provide addressability to the input parameter.

On entry to the user exit, register 1 points to the following parameter list:

Offset	Description
0	A pointer to a mixed INET address
4	A pointer to JOB statement parameters
8	A pointer to EXEC statement parameters
12	A pointer to an optional JES control statement

The INET address consists of the following fields:

Offset	Description
0	2 bytes (AF_INET or 2)
2	2 bytes (server port)
4	4 bytes (client INET address)

The JOB statement parameters can be up to 1024 characters in length and are ended by X'00'. You can modify the parameters with the exit routine. Upon entry, the parameters are set to:

- *user_ID*
- USER=*user_id*
- PASSWORD=*password*
- MSGCLASS=*msgclass*

MSGCLASS is as specified in the Remote Execution cataloged procedure. *userid* and PASSWORD are as received from the requesting client.

For RSH commands without passwords, note that the PASSWORD= parameter is not present. The *userid* in the first positional parameter can be processed by an installation-written JES exit.

The EXEC statement parameters can be up to 256 bytes in length and are ended by X'00'. These parameters can be modified by the exit routine. On entry, it contains the EXEC statement for the procedure specified in the TSOPROC parameter of the Remote Execution server or the default IKJACCNT procedure if TSOPROC is not specified.

The JES control statement parameter can be up to 256 bytes in length and is ended by X'00'. Upon entry, the parameter field is set to X'00'. Any JES control statement added by the user exit will be put between the JOB and the EXEC statement.

The modified JOB and EXEC statements are submitted as a TSO batch job.

The following user exit is shipped as a sample in the RXUEXIT member of the SEZAINST data set.

```
*********************************************************************
*
*             TCP/IP for MVS
*
* Name:       RXUEXIT
*
* SMP/E Distribution Name: EZAEBRXU
*
* Function:   This exit will add a CLASS parameter to the JOB
*             statement submitted by the REXECD server in MVS
*             TCP/IP.
*
* Copyright:  Licensed Materials - Property of IBM
*             This product contains "Restricted Materials of IBM"
*             5655-HAL (C) Copyright IBM Corp. 1994.
*             All rights reserved.
*             US Government Users Restricted Rights -
*             Use, duplication or disclosure restricted by
*             GSA ADP Schedule Contract with IBM Corp.
*             See IBM Copyright Instructions.
*
* Interface:  R1 -> input parameter list:
*             +0  -> AF-INET Socket address structure of
*                    REXEC Client
*             +4  -> JOB statement buffer
*             +8  -> EXEC statement buffer
*             +12 -> JES control statement buffer
*
* Logic:      The typical contents of the JOB statement buffer
*             are; userid,USER=userid,PASSWORD=password,
*             MSGCLASS=H
*
*             The JOB statement buffer is 1024 bytes in length.
*             The contents of the buffer are null terminated.
*             If the buffer contents are altered, the user must
*             ensure they are null terminated (one byte x'00')
*             and that the total length including termination
*             byte does not exceed the buffer length.
*
*             The JES control statement buffer is 256 bytes in
*             length and the contents are null terminated.
*
* Abends:     - none -
*
```

```
* Returncode: RC = 0
*
*********************************************************************
PARMS    DSECT
PTRINET  DC    F'0'                *-> AF-INET socket address
PTRJOBP  DC    F'0'                *-> Job statement parameters
PTREXECP DC    F'0'                *-> EXEC statement parameters
PTRJES   DC    F'0'                *-> EXEC statement parameters
*
BUFSIZE  EQU   1024                *JOB statement buffer size
*
.* RXUEXIT INIT   'REXECD add class parameter to JOB statement'
.* *
RXUEXIT  CSECT                     Establish the RXUEXIT csect
         USING RXUEXIT,12          Establish code addressability
         STM   14,12,12(13)        Save the caller's registers
         LR    12,15               Setup the local base register
         LR    2,1                 *Parm pointer
         USING PARMS,2             *Parameter addressability
         L     4,PTRJOBP           *-> Job card parameters
         LR    5,4                 *-> Start of buffer
         LA    6,1                 *Scan 1 byte at a time
         LA    7,BUFSIZE(5)        *-> First byte after buffer
         BCTR  7,0                 *-> Last byte to scan
SCANLOOP EQU   *
         CLI   0(5),0              *Is this string termination ?
         BE    GOTEND              *- Yes
         BXLE  5,6,SCANLOOP        *Continue scan for term
* ---------------------------------------------------------------
* If string is not null terminated, return without altering
* ---------------------------------------------------------------
         B     RETURN              *Should not happen.
GOTEND   EQU   *
         LA    5,1(5)              *incl. x'00' terminator
         LR    6,5                 *address of null byte
         SR    5,4                 *L'job parameter statements
         LA    5,L'CLASS(5)        *New parameter length
         CH    5,=AL2(BUFSIZE)     *Do we exceed buffer size?
         BNH   LENOK               *- No, there is room enough
* ---------------------------------------------------------------
* String length would exceed buf size so return without altering
* ---------------------------------------------------------------
         B     RETURN              *Return without modification
*
LENOK    EQU   *
         MVC   0(L_CLASS,6),CLASS  *Move class statement to buff
         L     6,PTRJES            *Get address of JES buffer
         MVC   0(L_JES2,6),JES2CNTL *Move JES2 control to buffer
RETURN   EQU   *
         LM    14,12,12(13)        Restore the registers
         LA    15,0(0,0)           Load the return code
         BR    14                  Return
*
         LTORG
*
CLASS    DC    C',CLASS=A'         *Class statement
         DC    X'00'               *null termination byte
L_CLASS  EQU   *-CLASS
*
JES2CNTL DC    C'/*JOBPARM SYSAFF=SYSA' *JES2 system affinity
         DC    X'00'                    *null termination byte
L_JES2   EQU   *-JES2CNTL
*
JES3CNTL DC    C'//*MAIN SYSTEM=(MAIN1)' *JES3 main assignment
```

```
         DC    X'00'                    *null termination byte
L_JES3   EQU   *-JES3CNTL
*
         END
```

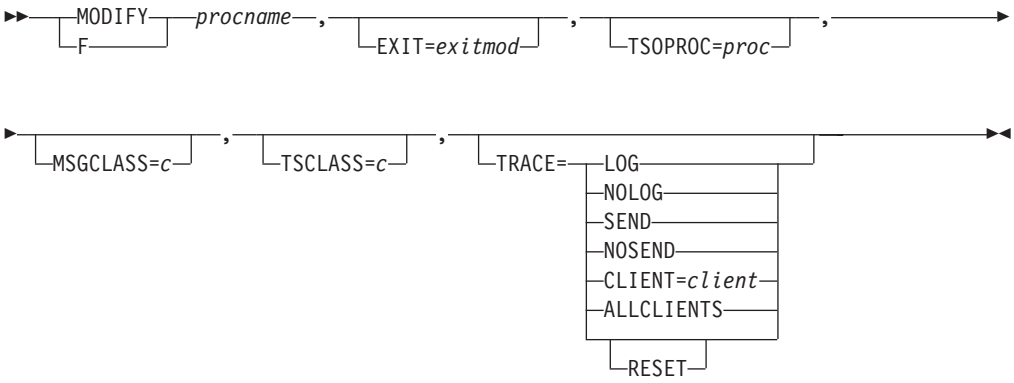
Modifying the Remote Execution Server Operating Parameters

You can update the Remote Execution server operating parameters during execution with the MODIFY command. All but MAXCONN can be changed.

MODIFY Command—Remote Execution Server

Use the MODIFY command to change the parameters on the Remote Execution server.

Syntax



Parameters

For a description of the valid parameters, see “Specify the Remote Execution Server Parameters” on page 628.

Examples

To change the user exit and TSO batch procedure, you might enter:

```
F RXSERVE,EXIT=USERX22,TSOPROC=KHFLACCN
```

Usage Notes

You cannot use the MODIFY command to change the MAXCONN parameter.

Chapter 19. Configuring the SMTP Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

Note: Before configuring the SMTP server, it is assumed that the necessary SYS1.PARMLIB change have been made. Consult the Program Directory for current information about the storage estimates for this version. The Program Directory also contains information about customization of certain SYS1.PARMLIB members, which must be completed before the initial program load (IPL) for the MVS image.

This chapter describes how to configure the Simple Mail Transfer Protocol (SMTP) server. There is also a section about operating the SMTP server.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Configuration Process

Steps to configure SMTP:

1. Specify AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set.
2. Update the SMTP cataloged procedure *hlq*.SEZAINST(SMTPPROC).
3. Customize the SMTPNOTE CLIST and modify PARMLIB members.
4. Customize the SMTP mail headers (optional).
5. Set up a TCP-to-NJE mail gateway (optional).
6. Specify configuration statements in the SMTP configuration data set.
7. Create an SMTP security table (optional).
8. Enable SMTP domain name resolution.

Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP

If you want the SMTP server to start automatically when the TCPIP address space is started, include the name of the member containing the SMTP catalogued procedure in the AUTOLOG statement of the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  SMTP
ENDAUTOLOG
```

To ensure that port TCP 25 is reserved for SMTP, verify that the name of the member containing the SMTP cataloged procedure has been added to the PORT statement in *hlq*.PROFILE.TCPIP.

```
PORT
  25 TCP SMTP
```

For more information on the AUTOLOG statement, see “AUTOLOG Statement” on page 137 . For more information on the PORT statement, see “PORT Statement” on page 229.

Step 2: Update the SMTP Cataloged Procedure

Update the SMTP cataloged procedure by copying the sample in *hlq*.SEZAINST(SMTPPROC) to your system or recognized PROCLIB. Specify SMTP parameters, and change the data set as required to suit your local configuration.

SMTP Cataloged Procedure (SMTPPROC)

This procedure contains the data set name for the SMTP configuration data set.

```
//SMTP PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE/',SYSERR=SYSERR
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEB017
//*
//*        5655-HAL (C) Copyright IBM Corp. 1989, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//* Change Activity:
//* $P1=MV11439 HTCP310 RTPMCL: Added "NOSPIE" to PARMS list.     @P1A
//*
//* Turn on SMSG support
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SMTP     EXEC PGM=MVPMAIN,
//             PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS',
//             REGION=6144K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSPRINT points to a data set used for the output from
//*        internal calls to IDCAMS.  It can be a temporary data set.
//*
//SYSPRINT DD SYSOUT=*
//*
//*        SYSERR contains runtime diagnostics from Pascal.  It can be
//*        a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*        SYSDEBUG receives output that is generated when the DEBUG
//*        configuration statement is specified in SMTP.  It can be
//*        a data set or SYSOUT.
//*
//SYSDEBUG DD SYSOUT=*
//*
//*        OUTPUT contains the startup and shutdown messages from SMTP.
//*        It can be a data set or SYSOUT.
//*
//OUTPUT   DD SYSOUT=*
//*
//*        LOGFILE receives output that is generated when the LOG
```

```
//*        configuration statement is specified in SMTP.  It can be
//*        a data set or SYSOUT.
//*
//LOGFILE  DD SYSOUT=*
//*
//*        SMTPNJE is the output of the SMTPNJE command.
//*        Before running SMTP you should use the SMTPNJE command
//*        to create the data set and then you can remove the
//*        "*" from the following line to activate SMTPNJE.
//*
//*SMTPNJE  DD DSN=TCPIP.SMTPNJE.HOSTINFO,DISP=SHR
//*
//*        CONFIG points to a sample configuration data set.
//*        Before running SMTP you should modify this file to
//*        include parameters suitable for your installation.
//*        Refer to the Chapter "Configuring the SMTP Server"
//*        in the Customization and Administration Guide.
//*
//CONFIG   DD DSN=TCPIP.SEZAINST(SMTPCONF),DISP=SHR
//*
//*        SECTABLE points to your SMTP security table data set.
//*        If you are running with the SECURE option, this data set
//*        will contain a list of NJE users who are authorized to
//*        use the gateway.  Refer to the Chapter "Configuring the
//*        SMTP Server" in the Customization and Administration Guide.
//*        You must remove the "*" from the following line to allow
//*        SMTP to find the data set.
//*
//*SECTABLE DD DSN=SMTP.SMTP.SECTABLE,DISP=SHR
//*
//*        SMTPRULE points to the data set containing the rewrite rules
//*        for the header addresses. You must specify REWRITE822HEADER
//*        YES for this data set to be read. Refer to the Chapter
//*        "Configuring the SMTP Server" in the Customization and
//*        Administration Guide.  You must remove the "*" from the
//*        following line to allow SMTP to find the data set.
//*
//*SMTPRULE DD DSN=SMTP.SMTP.RULE,DISP=SHR
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task. The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 3: Customize the System CLIST and Modify PARMLIB Data Sets

You must copy and customize the SMTPNOTE CLIST on every system where users will be able to send mail with the SMTPNOTE command. This includes TCP/IP nodes and each NJE node that sends mail through SMTP on a remote gateway node. SMTPNOTE uses the TSO transmit (XMIT) command to interface with SMTP.

Copy the SMTPNOTE member of the *hlq*.SEZAINST data set into the system CLIST data set. Since the *hlq*.SEZAINST data set is in a fixed format, the SMTPNOTE member may be truncated if your system CLIST library is not in a fixed format.

You should customize the following variables in the SMTPNOTE CLIST:

HOSTNAME
The name of the system on which this CLIST is installed. Typically, the name is the NJE node name of this system.

SMTPNODE
The NJE node on which the SMTP server runs. Typically, HOSTNAME and SMTPNODE have the same value. When SMTPNODE is used on an NJE network in conjunction with a TCP-to-NJE gateway, the value of this parameter is the NJE node name of that gateway.

SMTPJOB
The name of the address space in which SMTP runs at SMTPNODE. Usually this is SMTP.

TEMPDSN
The name of the temporary data set used to store the contents of the note being created. This can be any arbitrary data set name that ends with the low-level qualifier, TEXT. Do not use a fully qualified name. If you do not fully qualify the name (no quotes), the data set name will be prefixed by the *userid*. If you enclose the name in single quotes, several users can use this temporary data set.

You should also modify the following PARMLIB members:

IEFSSNxx
The IEFSSNxx member may be modified in one of the following two ways:

- The following lines may be included:

```
TNF,MVPTSSI
VMCF,MVPXSSI, nodename
```

  where *nodename* is the NJE node name. The NJE node name, *nodename*, must be the same as the *hostname* and the *smtpnode* in the SMTPNOTE CLIST.
- If you are using restartable VMCF, you must make changes to IEFSSxx members in the SYS1.PARMLIB data set.

  For introductory information on restartable VMCF, refer to “Step 3: Configuring VMCF and TNF” on page 37. For the MVS system changes required for restartable VMCF, refer to the TCP/IP for MVS Program Directory. For information on VMCF commands refer to *OS/390 SecureWay Communications Server: IP Diagnosis*.

IKJTSOxx
The TRANSREC statement must contain the correct nodename.

Step 4: Customize the SMTP Mail Headers (Optional)

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 contents. Envelopes contain all necessary information to accomplish transmission and delivery of the message content. The fields within the envelope are in a standard format.

In most cases, the mail header defaults are adequate. If it is necessary for you to change them, you can use the REWRITE822HEADER statement in the SMTP configuration data set to control the way SMTP performs a rewrite of the RFC 822 mail headers. Mail headers are passed from the local system, or NJE network, to the TCP network. Mail headers passing from the TCP network to the local system

or NJE network are 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:

Field **Description**

From Is the identity of the person sending the message.

Resent-From
Indicates 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
Indicates the person to whom you should forward the reply.

Return-Path
This field is added by the mail transport service at the time of final delivery. It contains definitive information about the address and route back to the originator of the message.

Sender
Is the authenticated identity of the agent that sent the message. An agent can be a person, system, or process.

Resent-Sender
Is 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 contents of this field are not included in copies sent to the primary and secondary recipients of the message but are included in the author's copy.

The SMTP Rules Data Set

You can override the default rules for header addresses by creating a SMTP rules data set. This allows you to customize the address transformations to the needs of a particular site. If you are customizing SMTP mail headers, this task is required.

The SMTP rules data set is pointed to by the //SMTPRULE DD statement in the SMTP cataloged procedure. The SMTP rules data set consists of:

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

Statement Syntax: In creating the SMTP rules data set you must use the following syntax conventions:

- The data set 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 end every statement with a semicolon (;).

- A character string appearing within single quotation marks ('...') is not case-sensitive. For example, 'abc' represents 'abc', 'Abc', 'ABC', and so forth. 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 data set.
- Format of the field definition section
- Format of the rules definition section

Format of the Field Definition Section: The field definition section is the first section in any SMTP rules data set. 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 the following). 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***, and ***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 contains 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 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: Certain field names can be added to or omitted from 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 2 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 sets:

```
HerAlias   = 'Reply-To' 'Sender';
HisAlias   = TheirAlias - 'Warnings-To' - 'Bcc' + HerAlias;
```

In the previous example, `HisAlias` is an alias that represents the following fields:

```
HisAlias:  'To' 'From' 'Cc' 'Errors-To' 'Reply-To' 'Sender'
```

Intersection Operations: 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 data set. It contains the header rewriting rules that define the intended address transformations, and it is introduced by the following heading:

```
Rule Definition Section
```

The basic form of a rewrite rule is:

```
alias :before-address-pattern  => after-address-pattern;
```

where 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 '@' NJEHostName => A '@' TCPHostName; -- convert host names
```

In the previous example, `A '@' NJEHostName` 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 NJE host name (`NJEHostName`) 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`.

SMTP Rules Syntax Conventions: Use the following syntax convention when writing SMTP rules:

- Some keywords have special meaning to the rules interpreter. For example, `NJEHostName` keyword means the NJE 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 644. 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 incorrect 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, which is not valid.

  ```
  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 `AltNJEDomain` defined in the system configuration data set, no rule that includes `AltNJEDomain` 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 NJE 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, within the

before-address-pattern of a given rule, no two wildcards are allowed in a row, and the same wildcard cannot be used more than once. The following rules have valid syntax:

```
A '@' B TCPHostName      => A '%' B '@' TCPHostName;
A '%' B '@' NJEHostName => A B '@' TCPHostName;
```

The following rules have incorrect syntax because the first rule has 2 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 '@' NJEHostName => 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 NJE 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 strings that are not case-sensitive, 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 data set 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 ended by an ENDIF keyword. As with IF statements in other programming languages, the ELSE clause is optional.

There are only two conditions recognized by an IF statement:
1. IF *predefined keyword* = '*character string*' THEN ... ENDIF
2. 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 example shows the use of IF statements.

```
IF NJEDomain = ' THEN
   A '@' AnyNJEHostName => A '%' AnyNJEHostName '@' TCPHostName;
ELSE
```

```
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                  =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                   =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
   ENDIF
ENDIF
```

Predefined Keywords within the SMTP Rules

The following predefined keywords can be used to define the header rewriting rules:

AltNJEDomain
Matches the alternative domain name of the NJE network as defined by the ALTNJEDOMAIN statement in the SMTP configuration data set.

AltTCPHostName
Matches any alternative TCP host name of the system, as defined by ALTTCPHOSTNAME statements in the SMTP configuration data set.

AnyDomainName
Matches any fully qualified domain name. Any host name with a period (.) is considered to be a fully qualified domain name.

AnyNJEHostName
Matches any (unqualified) NJE host name defined in the SMTPNJE.HOSTINFO data set.

NJEDomain
Matches the domain name of the NJE network as defined by the NJEDOMAIN statement in the SMTP configuration data set.

NJEHostName
Matches the NJE host name of the system.

SecureNickAddr
Matches an address of the form *NJE_user_id@NJE_node_id*, where *NJE_user_id*, and *NJE_node_id* are defined with a nickname in the SMTP security data set.

Note: This only matches 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 security data set. When SecureNickName is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickAddr with the

corresponding *NJE_user_id@NJE_node_id*. This allows SecureNickAddr to be specified in the *after-address-pattern*.

ShortTCPHostName
Matches the first portion of the TCP host name of the system, as defined by the HOSTNAME statement in the *hlq*.TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of ShortTCPHostName is `mvs1`.

TCPHostName
Matches the TCP host name of the system as defined by the concatenation of the HOSTNAME and DOMAINORIGIN statements in the *hlq*.TCPIP.DATA data set.

TCPHostNameDomain
Matches the domain portion of the TCP host name of the system as defined by the DOMAINORIGIN statement in the *hlq*.TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of TCPHostNameDomain is `acme.com`.

The predefined keywords 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 //SMTPRULE DD statement is not found, SMTP uses a default set of rules. The default set used depends on whether SMTP is configured as a secure gateway.

SMTP Nonsecure Gateway Configuration Defaults: If SMTP is not configured as a secure gateway, SMTP uses the following defaults:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section


A '@' NJEHostName => A '@' TCPHostName;

IF NJEDomain = ' THEN
   A '@' AnyNJEHostName => A '%' AnyNJEHostName '@' TCPHostName;
ELSE
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                  =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                   =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
```

```
         A '%' AnyNJEHostName '.' NJEDomain '@' 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 defaults:

```
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 '@' NJEHostName => A '@' TCPHostName;

IF NJEDomain NOT = ' THEN
   SecureNickAddr '.' NJEDomain    => SecureNickName '@' TCPHostName;
   SecureNickAddr '.' AltNJEDomain => SecureNickName '@' TCPHostName;
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                   =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' 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 examples show how the header rewriting rules affect an SMTP mail header. The example site is not a secure gateway and is configured as follows:

```
   TCPHostName      = mvs1.acme.com
   ShortTCPHostName = mvs1
   AltTCPHostName   = seeds.acme.com
   NJEHostName     = mvs1
   NJEDomain       = acmenet
   AltNJEDomain    = centralnet
```

Note that the above keywords are configured according to the definitions found in “Predefined Keywords within the SMTP Rules” on page 644 (for example, from *hlq*.TCPIP.DATA). In addition, assume that the following are known to be other NJE hosts:

```
bird
iron
```

Then the following header:

```
From: abc@mvs1 (Brendan Beeper)
To: Jenny Bird <def@bird>
Cc: ghi@iron.acmenet, j@mvs1,
 k@seeds.acme.com,
 Mailing List <owner@acmenet>,
 lmno@iron.centralnet
Subject: New Ore
```

is rewritten by the default header rewriting rules as:

```
From: abc@mvs1.acme.com (Brendan Beeper)
To: Jenny Bird <def%bird.acmenet@mvs1.acme.com>
Cc: ghi%iron.acmenet@mvs1.acme.com, j@mvs1.acme.com,
 k@mvs1.acme.com,
 Mailing List <owner%acmenet@mvs.acme.com>,
 lmno%iron.acmenet@mvs1.acme.com
Subject: New Ore
```

The next example deviates from the defaults listed in “Default SMTP Rules” on page 645. On the configuration for nonsecure gateways, if you change the rule before the 2 ENDIFs to:

```
A '@' AnyNJEHostName '.' AltNJEDomain =>
   '<@' TCPHostName ':' A '@' AnyNJEHostName '.' NJEDomain '>';
```

then the last address in the Cc: field within our header is rewritten as:

```
Cc: <@mvs1.acme.com:lmno@iron.acmenet>
```

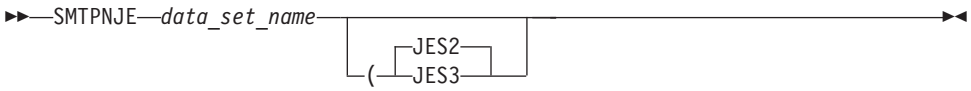
Note: Do not make the change shown in the previous example; it is intended only as a demonstration of the capabilities of the pattern-matching language.

Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)

You can configure the SMTP server to run as a mail gateway between TCP network users and users located on a NJE network attached to the local host. This way NJE users can send mail or data sets to users on TCP hosts using SMTPNOTE. See *OS/390 SecureWay Communications Server: IP User's Guide* for more information about SMTPNOTE. For JES2, see *MVS/ESA JES2 Initialization and Tuning* , SC28-1038-1. For JES3, see *MVS/ESA JES3 Initialization and Tuning* , SC23-0073-2.

Follow these steps to set up your TCP-to-NJE mail gateway:

1. Add the GATEWAY statement to the SMTP configuration data set. Add other related statements, such as ALTDOMAIN, NJECLASS, NJEDOMAIN, and NJEFORMAT, as required by your configuration.
2. Issue the SMTPNJE command.



data_set_name
The name of the input data set for SMTPNJE. It specifies the initialization data set of the JES2 or JES3 subsystem that is scanned for NJE nodes by SMTPNJE. The data set name is the same name as defined on ddname HASPPARM in your JES2 procedure or in the JES3IN ddname in your JES3 procedure.

(Required delimiter.

JES2 or JES3
Denotes whether the initialization data set being pointed to is for JES2 or JES3. If omitted, the default is JES2. For JES2, the SMTPNJE program scans for the keywords NODE and DESTID from which it extracts the information. For JES3, the keyword scanned for is NJERMT.

The SMTPNJE program creates the NJE host table data set called *user_id*.SMTPNJE.HOSTINFO. You can rename this data set and include the name of the data set on the SMTPNJE DD statement in the SMTP cataloged procedure. The //SMTPNJE DD statement is required.

3. Install the SMTP server (along with the TCPIP address space) on the gateway node. Use the GATEWAY, NJEDOMAIN, and NJEFORMAT statements in the configuration data set. Optionally, you can use either the RESTRICT or the SECURE statements to limit which users can use the gateway.

Step 6: Specify Configuration Statements in SMTP Configuration Data Set

Copy the member *hlq*.SEZAINST(SMTPCONF) to your own SMTP configuration data set and modify it for your site using the SMTP configuration statements.

Note: If the SMTP configuration data set is a sequential data set, you can not edit the data set while SMTP is running. If the data set is a PDS member, it can be edited while SMTP is running.

Summary of SMTP Configuration Statements

The SMTP configuration statements are summarized in Table 24.

Table 24. Summary of SMTP Configuration Statements

Statement	***Description***	***Page***
ALTNJEDOMAIN	Specifies an alternative domain name of the NJE network, if SMTP is running as a mail gateway.	657
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.	658
BADSPOOLFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.	659
DBCS	Specifies that DBCS code conversion be performed on the mail.	660
DEBUG	Records all SMTP commands and replies.	663
FINISHOPEN	Specifies the SMTP wait time for connection.	664

Table 24. Summary of SMTP Configuration Statements (continued)

Statement	Description	Page
GATEWAY	Specifies operation of SMTP as a gateway.	665
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.	667
IPMAILERADDRESS	Specifies the IP address of an SMTP server that can resolve network addresses of unknown hosts.	668
LISTENONADDRESS	Allows you to restrict which IP address is used to receive mail on a multi-homed system.	669
LOCALCLASS	Specifies the spool data set class for local mail delivery.	670
LOCALFORMAT	Specifies the spool data set format for local host mail delivery.	671
LOG	Directs SMTP to log all SMTP traffic.	672
MAILER	Specifies the address of the batch SMTP server that receives mail.	673
MAILFILEDSPREFIX	Specifies the prefix to add to mail data sets.	675
MAILFILESUNIT	Specifies the unit where SMTP mail data sets reside.	676
MAILFILEVOLUME	Specifies the volume where newly allocated SMTP data sets reside.	677
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over a TCP connection.	678
NJECLASS	Specifies the spool data set class for mail delivered on an NJE network.	679
NJEDOMAIN	Specifies the domain name of the NJE network if SMTP functions as a gateway.	680
NJEFORMAT	Specifies the spool data set format for mail delivered on the NJE network.	681
NJENODENAME	Specifies the node name of the local JES2 or JES3 node for mail delivered on the NJE network.	682
NOLOG	Turns off the logging of mail transactions.	683
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP connections over which SMTP actively delivers mail.	684
PORT	Specifies an alternative port number for the SMTP server during testing.	685
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.	686
RCPTRESPONSEDELAY	Specifies how long the SMTP server delays responding to the RCPT commands.	687
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.	688
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.	689
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.	691
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP host.	692
REWRITE822HEADER	Prevents SMTP from rewriting RFC822 headers with source routing.	693

Table 24. Summary of SMTP Configuration Statements (continued)

Statement	Description	Page
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.	694
SMSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP SMSG commands.	695
SPOOLPOLLINTERVAL	Specifies the interval for SMTP to check the spool for incoming batch data sets.	696
TEMPERRORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.	697
TIMEZONE	Sets the 3-character printable name of the local time zone.	698
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender, indicating that the mail has so far been undeliverable and that SMTP will continue to retry delivery for RETRYAGE days.	699

Sample SMTP Configuration Data Set (SMTPCONF)

The following sample is found in *hlq*.SEZAINST(SMTPCONF).

```
;*********************************************************************
;                                                                    *
;   Name of Data Set:    SMTP.CONFIG                                 *
;                                                                    *
;   COPYRIGHT = NONE.                                                *
;                                                                    *
;   This data set is pointed to by the CONFIG DD statement in the    *
;   SMTP Cataloged Procedure (SMTPPROC).                             *
;                                                                    *
;   This data set is used to specify runtime options and data        *
;   to the SMTP server address space.                                *
;                                                                    *
;   Syntax Rules for the SMTP configuration data set:                *
;                                                                    *
;   (a) All characters to the right of and including a ; will be     *
;       treated as a comment.                                        *
;                                                                    *
;   (b) Blanks and <end-of-line> are used to delimit tokens.         *
;                                                                    *
;   See the TCP/IP for MVS:  Customization and Administration Guide  *
;   for a complete explanation of all the statements.                *
;                                                                    *
;*********************************************************************
;
;  Defaults that normally aren't changed:
;
PORT 25                 ; Port to accept incoming mail
BADSPOOLFILEID TCPMAINT ; Mailbox where unreadable spool files and
                        ; looping mail are transferred.
LOG                     ; Log all SMTP mail delivered
INACTIVE 180            ; Time-out for inactive connections
FINISHOPEN 120          ; Time-out for opening TCP connections
RETRYAGE 3              ; Keep retrying mail delivery for 3 days
WARNINGAGE 1            ; Warn sender that mail has been undeliverable
                        ; for 1 day, but that attempts to deliver the
                        ; mail will continue for another 2 days.
RETRYINT 20             ; Retry mail delivery every 20 minutes
MAXMAILBYTES 524288     ; Largest mail to accept over a TCP connection
RESOLVERRETRYINT 20     ; Retry pending name resolutions every 20 minutes
RCPTRESPONSEDELAY 60    ; How long to delay RCPT TO: response when
                        ; waiting for an address resolution.
TEMPERRORRETRIES 0      ; How many times to retry temporary delivery
```

```
                       ; errors.  The default, 0, means retry for
                       ; RETRYAGE days; otherwise the mail is returned
                       ; after this number of deliver attempts.
SPOOLPOLLINTERVAL 30   ; Amount of time in seconds between spool polling
TIMEZONE EST           ; Specifies the printable 3-letter name of
                       ; the local time zone.  Remember to change this
                       ; for daylight saving time.
; DEBUG                ; Normally not used, causes debug information to
                       ; be written to the SMTP DEBUG file.
;
;*********************************************************************
;
; The following statements tell SMTP where incoming mail is stored
; while it is being queued for delivery.
;
MAILFILEDSPREFIX SMTP   ; Data set prefix name for queued mail files
MAILFILEUNIT SYSDA      ; MVS unit name for new file allocations
; MAILFILEVOLUME volume ; MVS volume name for new file allocations
;
;*********************************************************************
;
; ALTTCPHOSTNAME is used to specify an alternative fully qualified host
; name by which SMTP will know the local host.  Mail sent to users at
; <hostname> is treated as if they were local users.
;
; ALTTCPHOSTNAME <hostname>
;
;*********************************************************************
;
; The POSTMASTER statement specifies the mailboxes where mail
; addressed to postmaster is spooled.  Multiple POSTMASTER statements
; can be specified when not running in SECURE mode.
;
POSTMASTER TCPMAINT
; POSTMASTER NJEuser@NJEnode
; POSTMASTER SMTPuser@SMTPnode
;
;*********************************************************************
;
; Use the SMSGAUTHLIST statement to specify the addresses of local
; users authorized to issue privileged SMTP SMSG commands.
;
SMSGAUTHLIST
    TCPMAINT
  ; OPERATOR  LocalUser
ENDSMSGAUTHLIST
;
;*********************************************************************
;
; The OUTBOUNDOPENLIMIT statement specifies the maximum number of
; simultaneous TCP connections over which SMTP will actively deliver
; mail.
;
; OUTBOUNDOPENLIMIT 30
;
;*********************************************************************
;
; Configuration for a typical NJE to TCP/IP mail gateway.
;
GATEWAY                ; Accept mail from and deliver mail to NJE hosts
NJEDOMAIN BITNET       ; Pseudo domain name of associated NJE network
NJEFORMAT PUNCH        ; NJE recipients receive mail in punch format
NJECLASS B             ; spool class for mail delivered by SMTP to the
                       ; NJE network
LOCALFORMAT NETDATA    ; Local recipients get mail in netdata format
                       ; Netdata allows TSO receive to be used with mail
LOCALCLASS B           ; Spool class for local mail delivered by SMTP
```

```
;
;*********************************************************************
;
; Use the ALTNJEDOMAIN to specify an alternate NJE domain name.
; This can be useful when the NJE network is known by multiple
; domain names, such as "vnet" and "vnet.ibm.com".
;
; ALTNJEDOMAIN vnet
;
;*********************************************************************
;
; The REWRITE822HEADER statement specifies whether SMTP will
; rewrite the RFC822 headers on all mail passing from NJE to TCP
; through the mail gateway.  The SMTP.RULES data set specifies how the
; server is to rewrite the headers.
;
; The PRINT | NOPRINT options specify whether SMTP will print
; the rules to the SMTP output when SMTP starts.
;
REWRITE822HEADER YES NOPRINT
;
;*********************************************************************
;
; Use MAILER to specify the address of a batch SMTP server to which
; SMTP delivers mail destined for various classes of recipients.  The
; old FOLDnoSOURCEroute parameter is equivalent to specifying PUNCH and
; NOSOURCEROUTES.
;
; LOCAL, NJE, and UNKNOWN specify conditions under which mail will
; be forwarded to the MAILER - see the Customization and Administration
; Guide for further details.
;
; MAILER MUSER@MNODE PUNCH NOSOURCEROUTES LOCAL NJE UNKNOWN
;
; MAILER ... UNKNOWN and IPMAILERADDRESS should not be used together.
;
; IPMAILERADDRESS x.xx.xxx.xx ; Routes mail sent to an unknown
;                      ; recipient to an SMTP server on an IP network
;
;*********************************************************************
;
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;
; RESTRICT RETURN          ; Return mail from restricted users
;   charming@ourvm.our.edu ; Don't accept any mail from Prince Charming
;   charming@OURVMX        ; via NJE or TCP network.
;   charming@ourvm*        ; This line takes the place of previous 2 lines!
;  *@castle                ; Don't accept mail from anyone at host castle
; ENDRESTRICT
;
;*********************************************************************
;
; Use the SECURE statement if this SMTP machine is to run as an SMTP-
; to-NJE Secure Gateway.  Only users in the SMTP.SECTABLE data set
; will be allowed to send mail, all other mail will be returned or
; rejected.  Note that the contents of dataset
; mailfiledsprefix.SECURITY.MEMO will be sent to NJE users that are
; not authorized to use the gateway.
;
; SECURE
;
;*********************************************************************
;
; Use the DBCS statement if this SMTP machine is to perform DBCS
; code conversion on the mail.  Consult the Customization and
; Administration Guide for the necessary parameters.
```

```
;
; DBCS   <parameters>
;
;
```

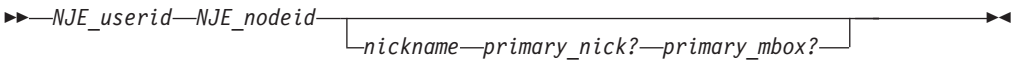
Step 7: Create an SMTP Security Table (Optional)

If you want to set up a secure TCP-to-NJE gateway, you need to:

- Include the SECURE statement in the SMTP configuration data set
- Create a security data set that contains a list of NJE users who are authorized to use the gateway
- Create a mailfiledsprefix.SECURITY.MEMO data set. The contents data set are sent to unauthorized NJE users whose mail is rejected. See Rejected Mail Examples for sample contents of this data set. This data set must be defined as LRECL=255 and RECFM=VB. It will be dynamically allocated by SMTP when needed.

The SMTP security data set is pointed to by //SECTABLE DD statement. It can have either fixed or variable length records of up to 255 characters in length. Records whose first nonblank character is an asterisk (*) are treated as comments and are ignored.

Use the following format when creating the list of NJE users:



NJE_userid
The NJE user ID of the authorized user.

NJE_nodeid
The NJE node ID of the authorized user.

nickname
The name by which this user is known on the TCP side of the gateway. This name must not contain any special characters, such as < > ( ) [ ] \ . , ; : @ and ".

primary_nick?
Either Y or N. If Y is specified, then mail addressed to *nickname*`@smtp-gateway` is automatically forwarded to *NJE_userid* `at` *NJE_nodeid*. Each nickname can have only one *primary_nick?* record set to Y.

primary_mbox?
Either Y or N. If Y is specified, then mail from *NJE_userid* `at` *NJE_nodeid* is converted to *nickname*`@smtp-gateway` before it is sent to the TCP recipient. Each *NJE_userid*, *NJE_nodeid* pair can only have one *primary_mbox?* record.

SMTP Security Data Set Examples

The following example shows an SMTP security data set:

```
*  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 NJE network addresses through the SMTP gateway is rewritten to the following TCP network addresses. Assume the host name of the gateway is SMTP-GATEWAY.IBM.COM.

```
NJE Address              TCP 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 network to the following TCP network addresses is forwarded to the following NJE network addresses. Assume the host name of the gateway is SMTP-GATEWAY.IBM.COM.

```
TCP Address                              NJE 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
SMITH%DALLAS@SMTP-GATEWAY.IBM.COM        SMITH at DALLAS
```

Rejected Mail Examples

SMTP rejects mail to or from an unauthorized NJE user. If the mail is from the TCP network, SMTP rejects the RCPT TO command with the error:

```
550 User is not a registered gateway user
```

If the mail is from the NJE network, SMTP rejects the MAIL FROM command with the error:

```
550 User is not a registered gateway user
```

and includes the *mailfiledsprefix*.SECURITY.MEMO data set as an explanation.

The following example shows a sample *mailfiledsprefix*.SECURITY.MEMO data set:

```
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 NJE user:

```
Date: Fri, 5 Jul 91 10:55:59 EST
From: SMTP@MVS1.ACME.COM
To: DANIEL@MVS1
Subject: Undeliverable Mail

MVS1.ACME.COM unable to deliver following mail to recipient(s):
    <MATT@SMTP-GATEWAY.IBM.COM>
MVS1.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, 5 Jul 91 10:55:56 EDT
From: <DANIEL@MVS1.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 NJE user:

```
Date: Fri, 5 Jul 91 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.
050 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@MVS1.ACME.COM>
DATA
Date: Fri, 5 Jul 91 11:34:17 EST
From: <MATT@SMTP-GATEWAY.IBM.COM>
To:   <DANIEL@MVS1.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
```

Step 8: Enable SMTP Domain Name Resolution

The SMTP server can be configured to use either a domain name server or the local site tables. The *hlq*.TCPIP.DATA data set must be configured before you can use the domain name resolution for SMTP.

To use a domain name server, configure the *hlq*.TCPIP.DATA data set with the IP address of one or more name servers. If the *hlq*.TCPIP.DATA data set does not point to any name servers, the local site tables are used. However, if the SMTP server is configured to use name servers, SMTP does not use the site tables.

When SMTP uses a domain name server, it asks the domain name server for the MX records for the host to which it is trying to connect. If SMTP does not find MX records for a host, it delivers mail only to the primary host listed in the A records. The MX and A records are coded in the domain name server database.

The basic idea behind MX records is to send the mail as close as possible to the final destination. The destination host may currently be inactive, for example, because it is in another time zone. SMTP needs a synchronous connection to

deliver the mail, but due to the different time zones, two systems might never be active at the same time. and would never be able to exchange mail.

Using MX records would allow the SMTP server to deliver the mail to an alternate host if the first one is unavailable. SMTP tries to deliver mail to the host with the lowest MX record count. If the host is not currently available, it tries the host with the next lowest count.

For example, if SMTP wants to send mail to USER@BASKET, it checks the name server for MX records and finds the following:

```
MVS20  BASKET  A
       BASKET  MX  0    MVS20
       BASKET  MX  5    MVS18
       BASKET  MX  10   VMQ
```

SMTP delivers the mail to the BASKET with the lowest count on its MX record. If `MVS20` is unable to receive the mail, SMTP then tries to deliver it to `MVS18`. If `MVS18` cannot receive the mail, it tries `VMQ`. If none of the hosts can receive the mail, SMTP stores the mail and queues it for later delivery, at which time the process repeats.

For more information about how to add MX records to your name server, consult RFC 974, “Mail Routing and the Domain System.”

To receive a detailed trace on how SMTP is resolving a particular host name, you can issue the SMSG SMTP TRACE command at the console. You can also add the TRACE RESOLVER statement when configuring the *hlq*.TCPIP.DATA data set, but this will also trace the name resolution for all the other applications using the name server. To prevent the console log from becoming too large, only use the TRACE RESOLVER statement for debugging.

SMTP Configuration Data Set Statements

These sections contain the data set statements for the SMTP configuration data set.

ALTNJEDOMAIN Statement

Use the ALTNJEDOMAIN statement to specify an alternative domain name of the NJE network when SMTP is running as a mail gateway.

Syntax

Parameters

domain
The alternative domain name of the NJE network. The alternative NJE domain name is a string of 1 to 64 alphanumeric characters.

Examples

Using the ALTNJEDOMAIN statement is especially helpful when the NJE network is known by multiple domain names, such as VNET and VNET.IBM.COM.

```
ALTNJEDOMAIN VNET
```

Usage Notes

The ALTNJEDOMAIN statement can be specified only once.

ALTTCPHOSTNAME Statement

Use the ALTTCPHOSTNAME statement to specify an alternative, fully qualified host name by which SMTP recognizes the local host. Mail sent to users at *host_name* are treated as if they were local users. You can use the ALTTCPHOSTNAME statement to specify up to 16 alternative host names.

Syntax

Parameters

host_name
The name of the destination host.

Examples

In this example, mail sent to users at PALACE are treated as if they were local users.

```
ALTTCPHOSTNAME PALACE
```

BADSPOOLFILEID Statement

Use the BADSPOOLFILEID statement to specify the user ID on the local system where SMTP transfers unreadable spool files and looping mail.

Syntax

Parameters

user_id
The user ID on the local system where bad spool files and looping mail are delivered. The user ID should be a maximum of 8 characters. The default is TCPMAINT. If RACF is active, then a RACF profile must be defined for this user ID.

Examples

In this example, unreadable spool files and looping mail are transferred to the user ID, DBARTON.

```
BADSPOOLFILEID DBARTON
```

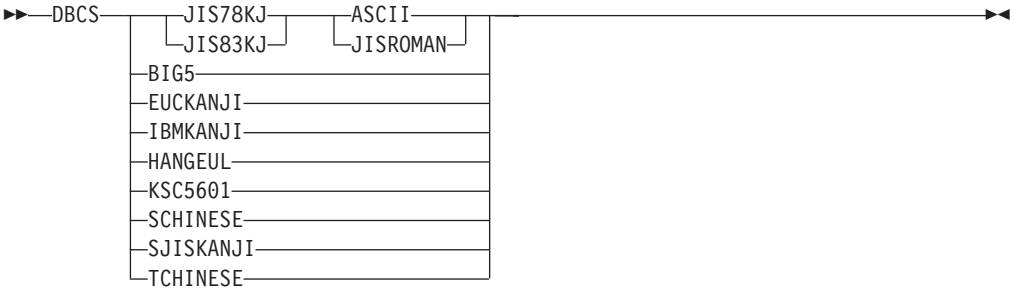
Usage Notes

The BADSPOOLFILEID statement can be specified only once.

DBCS Statement

Use the DBCS statement to specify that SMTP should perform DBCS code conversion on the mail. The parameters of the DBCS statement determine which translation table should be used in the conversion.

Syntax



Parameters

JIS78KJ
Specify JIS78KJ if the conversion between IBM Kanji and JIS 1978 DBCS codes is to be performed. The Escape Sequence for JIS X0208 1978 is `ESC 2/4 4/0`. SMTP will load the JIS78KJ DBCS translation table from the TCPKJBIN binary translate table data set.

Note: When JIS78KJ and JIS83KJ are used, either ASCII or JISROMAN must be used or an error will occur and SMTP will end; SMTP configuration will read the next parameter in the configuration file as the third DBCS statement entry.

JIS83KJ
Specify JIS83KJ if the conversion between IBM Kanji and JIS 1983 DBCS codes is to be performed. The Escape Sequence for JIS X0208 1983 is `ESC 2/4 4/2`. SMTP will load the JIS83KJ DBCS translation table from the TCPKJBIN binary translate table data set.

Note: When JIS78KJ and JIS83KJ are used, either ASCII or JISROMAN must be used or an error will occur and SMTP will end; SMTP configuration will read the next parameter in the configuration file as the third DBCS statement entry.

ASCII
Specify ASCII for JIS78KJ or JIS83KJ if the mail is shifted in ASCII code from JIS Kanji code. The Escape Sequence for ASCII is `ESC 2/8 4/2`.

JISROMAN
Specify JISROMAN for JIS78KJ or JIS83KJ if the mail is shifted in JISRoman code from JIS Kanji code. The Escape Sequence for JISRoman is `ESC 2/8 4/10`.

BIG5
Specify BIG5 if the conversion between IBM Traditional Chinese host DBCS codes and Big-5 PC DBCS codes is to be performed. SMTP will load the BIG5 DBCS translation table from the TCPCHBIN binary translate table data set.

EUCKANJI
Specify EUCKANJI if the conversion between IBM Kanji and Japanese EUC DBCS codes is to be performed. SMTP will load the EUCKANJI DBCS translation table from the TCPKJBIN binary translate table data set.

IBMKANJI
Specify IBMKANJI if IBM (EBCDIC) Kanji conversion is to be used. This option actually causes *no* conversion to be performed on the body of the mail. This may be used for the sending and receiving of mail in EBCDIC. If this option is selected, other SMTP servers on the same network should all be configured with IBMKANJI. 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.

The IBMKANJI transfer type does not require any translate table to be loaded.

HANGEUL
Specify HANGEUL if the conversion between IBM Korean host DBCS codes and Korean PC DBCS codes is to be performed. SMTP will load the HANGEUL DBCS translation table from the TCPHGBIN binary translate table data set.

KSC5601
Specify KSC5601 if the conversion between IBM Korean host DBCS codes and IBM KS DBCS codes is to be performed. SMTP will load the KSC5601 DBCS translation table from the TCPHGBIN binary translate table data set.

SCHINESE
Specify SCHINESE if the conversion between IBM Simplified Chinese host DBCS codes and Simplified Chinese PC DBCS codes is to be performed. SMTP will load the SCHINESE DBCS translation table from the TCPSCBIN binary translate table data set.

SJISKANJI
Specify SJISKANJI if the conversion between IBM Kanji and Shift JIS DBCS codes is to be performed. SMTP will load the SJISKANJI DBCS translation table from the TCPKJBIN binary translate table data set.

TCHINESE
Specify TCHINESE if the conversion between IBM Traditional Chinese host DBCS codes and Traditional Chinese 5550 PC DBCS codes is to be performed. SMTP will load the TCHINESE (5550) DBCS translation table from the TCPCHBIN binary translate table data set.

Examples

In this example, IBM Traditional-Chinese-to Traditional-Chinese 5550 PC code conversion will be used.

```
DBCS TCHINESE
```

Usage Notes

- The transmission of DBCS mail by SMTP actually uses 2 different translation tables; 1 SBCS and 1 DBCS. SBCS characters in the mail headers and in the mail body are converted using either *hlq*.STANDARD.TCPKJBIN, TCPHGBIN, TCPSCBIN, or TCPCHBIN.
- DBCS conversion is only performed on outgoing and incoming mail to and from other hosts. Mail spooled to SMTP (for example, via SMTPNOTE) for the local host, is delivered directly, without any DBCS code conversion.

Related Topics

“Chapter 36. Using Translation Tables” on page 1133

DEBUG Statement

Use the DEBUG statement to record SMTP commands and replies in the SMTP debug data set (which is pointed to by the SYSDEBUG DD statement).

Syntax

Parameters

None.

Usage Notes

The SMTP connection number is recorded along with each SMTP command or reply.

- Connection numbers 0 through MAXnumCONNECTIONS − 1 are used for SMTP connections over a TCP network.
- Connection number MAXnumCONNECTIONS + 1 is used for the batch SMTP connection.
- MAXnumCONNECTIONS has a default number of 256.

FINISHOPEN Statement

Use the FINISHOPEN statement to specify the number of seconds that SMTP waits while trying to establish a connection to a foreign site. After the specified number of seconds, SMTP ends the connection.

Syntax

Parameters

seconds
An integer in the range of 1 through 86400 indicating the number of seconds to wait for a connection to open. The default FINISHOPEN time-out is 120 seconds.

Examples

Set the time-out period to 90 seconds:

```
FINISHOPEN 90
```

GATEWAY Statement

Use the GATEWAY statement to have SMTP operate as a mail gateway between TCP network sites and NJE network sites (if the host system is connected to both a TCP network and an NJE network).

If you include the GATEWAY statement in the SMTP configuration data set, then SMTP will accept mail addressed to users on NJE hosts defined in the data set pointed to by the //SMTPNJE DD statement in the SMTP cataloged procedure. If you do not specify GATEWAY, SMTP rejects all mail that arrives from the NJE network or host.

Syntax

Parameters

None.

Examples

You can configure the SMTP server with the GATEWAY statement to run as a mail gateway between TCP network users and users located on a NJE network attached to the local host. Figure 25 illustrates this configuration.

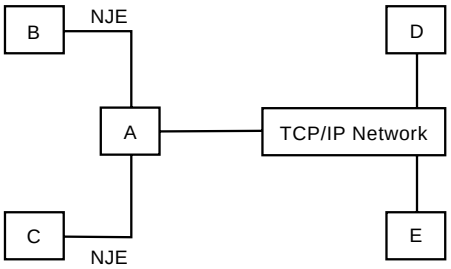


Figure 25. Example of a TCP-to-NJE Mail Gateway

In Figure 25:

- Host A is the local MVS host, running both TCP/IP and NJE.
- Hosts B and C are attached to host A through an NJE network.
- Hosts D and E are attached to host A through a TCP network.

Users on hosts A, B, and C can send mail or data sets to users on TCP hosts D and E using SMTPNOTE.

Usage Notes

If you do not include the GATEWAY statement in the SMTP configuration data set, SMTP will reject all mail that arrives from NJE.

Related Topics

- “Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)” on page 647
- “LOCALCLASS Statement” on page 670
- “LOCALFORMAT Statement” on page 671
- “NJECLASS Statement” on page 679
- “NJEDOMAIN Statement” on page 680
- “NJEFORMAT Statement” on page 681

INACTIVE Statement

Use the INACTIVE statement to specify the number of seconds of inactivity before SMTP will consider a connection to be inactive and close the connection.

Syntax

Parameters

seconds
An integer in the range of 1 through 86400 that specifies the number of seconds, after which SMTP considers the connection to be inactive. The default inactivity time-out is 180 seconds.

Examples

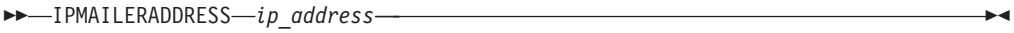
Set the seconds of inactivity allowable to 90 seconds:

```
INACTIVE 90
```

IPMAILERADDRESS Statement

Use the IPMAILERADDRESS statement to re-route mail that was sent to an unknown host and direct it to an SMTP server on an IP network rather than to a user on a local or NJE network. The SMTP server should have network connectivity and be able to perform name resolution.

Syntax



Parameters

ip_address
The dotted decimal address of an SMTP server on an IP network.

Examples

In this example, 7.89.256.72 is the address of the SMTP server on an IP network.

```
IPMAILERADDRESS 7.89.256.72
```

Usage Notes

IPMAILERADDRESS and MAILER... UNKNOWN provide the same function and should not be used together.

Related Topics

"MAILER Statement" on page 673

LISTENONADDRESS Statement

Use the LISTENONADDRESS statement to restrict which IP address will receive mail on a multi-homed system.

Syntax

Parameters

ip_address
The dotted decimal address of an SMTP server on an IP network.

Examples

In this example, 7.89.256.72 is the address of the SMTP server on an IP network that will be the home address for mail.

```
LISTENONADDRESS 7.89.256.72
```

Related Topics

"MAILER Statement" on page 673

LOCALCLASS Statement

Use the LOCALCLASS statement to specify the spool class for local mail delivered by SMTP.

Syntax

Parameters

class
The default is B (normally a punch class).

Examples

Set the spool class for local mail delivered by SMTP:

```
LOCALCLASS B
```

Usage Notes

The value used in this statement is site-dependent. Before setting this class, check with your system administrator for the site-dependent information. We recommend using the punch class of your system.

Related Topics

- “Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)” on page 647
- “GATEWAY Statement” on page 665
- “LOCALFORMAT Statement” on page 671
- “NJECLASS Statement” on page 679
- “NJEDOMAIN Statement” on page 680
- “NJEFORMAT Statement” on page 681

LOCALFORMAT Statement

Use the LOCALFORMAT statement to specify the spool data set format for mail delivered to users on the local host.

Syntax

Parameters

NETDATA
For NETDATA format, records can be longer than 80 characters and arrive as message-type records. The data set name is the first 8 characters of the sender's user ID.

NETDATA is the default format.

PUNCH
For PUNCH format, records are folded up to 80 characters in length or less. The spool data set is in NATIVE PUNCH format. The data set name is the first 8 characters of the sender's user ID.

Examples

Set the spool format for local mail delivered by SMTP:

```
LOCALFORMAT NETDATA
```

Usage Notes

It is recommended that you use the default value of NETDATA since the TSO RECEIVE command will indicate that it has an invalid file with PUNCH format output.

Related Topics

- "Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)" on page 647
- "GATEWAY Statement" on page 665
- "LOCALCLASS Statement" on page 670
- "NJECLASS Statement" on page 679
- "NJEDOMAIN Statement" on page 680
- "NJEFORMAT Statement" on page 681

LOG Statement

Use the LOG statement to log all SMTP traffic. The origin, sender, and recipients of each piece of mail are written to a log.

Syntax

Parameters

None.

Usage Notes

The log information goes to the data set specified on the //LOGFILE DD statement of the SMTP cataloged procedure. If no //LOGFILE DD statement is included in the cataloged procedure, information is not logged.

If neither LOG or NOLOG is specified in the SMTP configuration data set, the default is LOG.

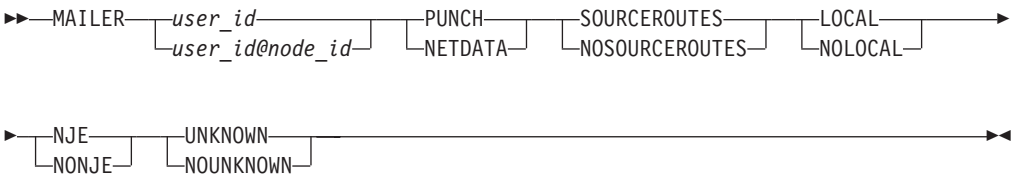
Related Topics

"NOLOG Statement" on page 683

MAILER Statement

Use the MAILER statement to specify the address of a batch SMTP server to which SMTP delivers mail destined for various classes of recipients.

Syntax



Parameters

user_id
Specifies the user ID of the local MAILER server.

user_id@node_id
Specifies the NJE address of the MAILER server.

PUNCH
Specifies that the MAILER server can only accept punch format spool data sets. Batch SMTP header records longer than 80 characters are split and an EBCDIC new line character (hex 15) is placed in column 80 to indicate that the record is continued. Records within the body of the mail that are longer than 80 characters are split across multiple punch records.

NETDATA
Specifies that the MAILER server accepts NETDATA format spool data sets. The NETDATA protocol automatically handles records longer than 80 characters.

SOURCEROUTES
Specifies that the MAILER server accepts BSMTP header addresses with source routes.

A source route contains routing information as well as the mailbox information. An example of a source route address is:

```
@host1,@host2:userid@host3.
```

The mailbox information in this example is `userid@host3.`

NOSOURCEROUTES
Specifies that the MAILER server does not accept source routes in the BSMTP header addresses.

Specifying NOSOURCEROUTES indicates that the address strings must be mailbox information only.

LOCAL
Specifies that mail for local recipients is spooled to the MAILER server.

NOLOCAL
Specifies that mail for local recipients is spooled directly to the recipients.

NJE
Specifies that mail for recipients on the NJE network is spooled to the MAILER server.

NONJE
Specifies that mail for recipients on the NJE network is spooled directly to the recipients.

UNKNOWN
Specifies that mail for recipients on an unknown host is spooled to the MAILER server.

NOUNKNOWN
Specifies that mail for recipients on unknown hosts is returned to the sender as undeliverable.

Examples

Use the MAILER option if you run with the Columbia Mailer.

```
MAILER MUSER@MNODE PUNCH NOSOURCEROUTES LOCAL NJE UNKNOWN
```

Usage Notes

- The MAILER server must either have a local address or be on the associated NJE network. The MAILER statement has no defaults; you must specify the parameters you want to use.
- IPMAILERADDRESS and MAILER... UNKNOWN provide the same function and should not be used together.
- All MAILER statement parameters must be specified or an error will occur and SMTP will terminate; eliminating a parameter will cause SMTP configuration to read the next statement in SMTPCONF as part of the mailer statement.

Related Topics

"IPMAILERADDRESS Statement" on page 668

MAILFILEDSPREFIX Statement

Use the MAILFILEDSPREFIX statement to specify the prefix that is added to the SMTP mail data sets. If multiple MVS systems share the same volume for SMTP mail data sets, specify a unique prefix qualifier for each SMTP server on MAILFILEDSPREFIX.

Data sets created with this prefix contain mail that is in the process of being received or delivered. Each piece of mail queued for delivery occupies a minimum of 2 tracks.

Syntax

Parameters

prefix
The prefix to add to the mail data sets. The prefix can be up to 20 characters in length, and a trailing period need not be specified. The default is the name of the job running SMTP.

Examples

Set the prefix name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEDSPREFIX SMTP
```

Usage Notes

All data sets are cataloged.

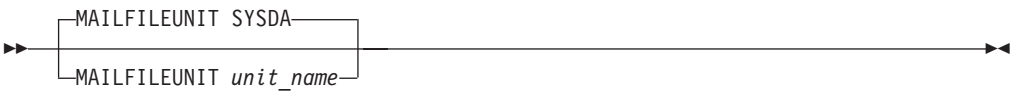
Related Topics

- “MAILFILEUNIT Statement” on page 676
- “MAILFILEVOLUME Statement” on page 677

MAILFILEUNIT Statement

Use the MAILFILEUNIT statement to specify the unit where the newly created SMTP mail data sets reside.

Syntax



Parameters

unit_name
The unit name where the data sets reside. The default is SYSDA.

Examples

Set the unit name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEUNIT SYSDA
```

Related Topics

- "MAILFILEDSPREFIX Statement" on page 675
- "MAILFILEVOLUME Statement" on page 677

MAILFILEVOLUME Statement

Use the MAILFILEVOLUME statement to specify the volume where newly allocated SMTP mail data sets reside.

Syntax

Parameters

volume_name
The volume name where the data sets reside. There is no default.

Examples

Set the volume name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEVOLUME volume6
```

Usage Notes

- If the volume name is not specified, SMTP allocates a storage volume.
- If your system does not have storage volumes, you must specify a volume name.

Related Topics

- “MAILFILEDSPREFIX Statement” on page 675
- “MAILFILEUNIT Statement” on page 676

MAXMAILBYTES Statement

Use the MAXMAILBYTES statement to specify the maximum size in bytes of mail that is accepted over a TCP connection.

Syntax

Parameters

bytes
The maximum number of bytes for incoming or outgoing mail. Mail arriving that is larger than this size, over a TCP connection, is rejected. The limits for this statement are 1 and 2147483647. The default size is 524288 (512KB) bytes.

Examples

Set the maximum size for mail to 32KB:

```
MAXMAILBYTES 32768
```

Usage Notes

The value used for *bytes* in the MAXMAILBYTES statement determines the space allocations for data sets allocated to hold the mail while it is being processed and is waiting for delivery. Be careful not to use too large a value or the data sets allocated will be too large.

NJECLASS Statement

Use the NJECLASS statement to specify the spool class for mail delivered by SMTP to the NJE network.

Syntax

Parameters

class
The spool class for mail delivered by SMTP. The default is B (which is normally a punch class).

Examples

Set the spool class for mail delivered to B:

```
NJECLASS B
```

Usage Notes

This statement is site-dependent. Before setting the class, check with your JES system administrator for site-dependent information. The recommended setting is the punch class of your system.

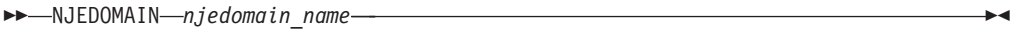
Related Topics

- “Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)” on page 647
- “GATEWAY Statement” on page 665
- “LOCALFORMAT Statement” on page 671
- “LOCALCLASS Statement” on page 670
- “NJEDOMAIN Statement” on page 680
- “NJEFORMAT Statement” on page 681

NJEDOMAIN Statement

Use the NJEDOMAIN statement to specify the domain name of the NJE network (if SMTP is running as a mail gateway). The NJE domain name is used to rewrite the header fields of mail passing from NJE network senders to TCP/IP network recipients.

Syntax



Parameters

njedomain_name
The NJE domain name. The default is a null string.

Examples

Set the NJE domain to BITNET:

```
NJEDOMAIN BITNET
```

Usage Notes

SMTP considers the NJE domain name BITNET to be a synonym for the European Academic Research Network (EARN and EARNET).

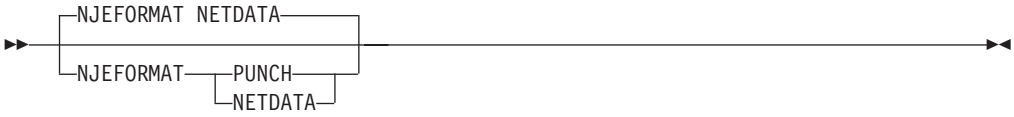
Related Topics

- "Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)" on page 647
- "GATEWAY Statement" on page 665
- "LOCALFORMAT Statement" on page 671
- "LOCALCLASS Statement" on page 670
- "NJECLASS Statement" on page 679
- "NJEFORMAT Statement" on page 681

NJEFORMAT Statement

Use the NJEFORMAT statement to specify the spool data set format for mail delivered to recipients on the NJE network.

Syntax



Parameters

PUNCH
Specifies that records are folded to 80 characters in length or fewer.

NETDATA
Specifies that records can be longer than 80 characters and that they arrive as MESSAGE-type records. The default format is NETDATA.

Examples

Set the format in which mail is sent to NJE recipients to PUNCH:

```
NJEFORMAT PUNCH
```

Usage Notes

This statement is valid only in GATEWAY mode.

Related Topics

- “Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)” on page 647
- “GATEWAY Statement” on page 665
- “LOCALFORMAT Statement” on page 671
- “LOCALCLASS Statement” on page 670
- “NJECLASS Statement” on page 679
- “NJEDOMAIN Statement” on page 680

NJENODENAME Statement

Use the NJENODENAME statement to specify the NJE node name of the local JES2 or JES3 node for SMTP. This statement overrides the value in the IEFSSN member and is an alternative to forcing the users to specify their real NJE node name in the IEFSSN member. It also allows the users to easily correct the name for SMTP's use, in case it was spelled wrong. Previously, they were required re-IPL to change the member because it was the only place from which SMTP would get the NJE node name. This value is not used in the place of the IEFSSN value as a selector in TCPIP.DATA.

Syntax

Parameters

node_name
The NJE node name of the local JES2 or JES3 node. The default is a null string.

Examples

Set the NJE node name to ALMADEN:

```
NJENODENAME ALMADEN
```

Related Topics

- "NJECLASS Statement" on page 679
- "NJEDOMAIN Statement" on page 680
- "NJEFORMAT Statement" on page 681

NOLOG Statement

Use the NOLOG statement to turn off logging information that indicates that mail has been received and delivered.

Syntax

Parameters

None.

Usage Notes

If neither LOG or NOLOG is specified in the SMTP configuration data set, the default is LOG.

Related Topics

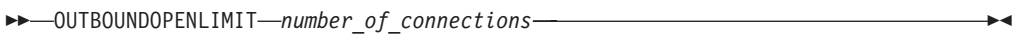
"LOG Statement" on page 672

OUTBOUNDOPENLIMIT Statement

Use the OUTBOUNDOPENLIMIT statement to specify a 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 delivery of mail. The OUTBOUNDOPENLIMIT statement should only be specified if there are limited TCP resources on the system and SMTP is using too many of these resources.

Syntax



Parameters

number_of_connections
A number in the range of 1 to the maximum number of TCP connections as defined for the user's system. If *number_of_connections* is out of range, the default, which is to impose no limits, is assumed.

Examples

Set the maximum number of simultaneous TCP connections to which mail is sent to 100.

```
OUTBOUNDOPENLIMIT 100
```

PORT Statement

Use the PORT statement to change the port reserved for SMTP. The port number 25 is normally reserved (in *hlq*.PROFILE.TCPIP) for the SMTP server to accept incoming mail requests.

Syntax

Parameters

port_num
An integer in the range of 1 through 65535 that specifies the port number to which SMTP listens. The default is 25.

Examples

Set the port for incoming mail to 1000:

```
PORT 1000
```

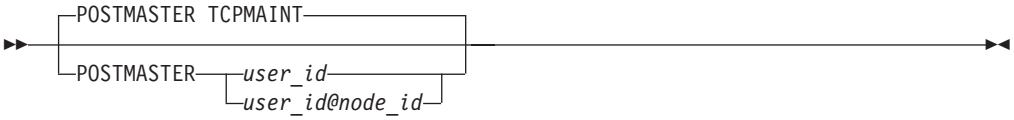
Usage Notes

- You can only specify a *port_number* if it has not already been reserved for some other server in *hlq*.PROFILE.TCPIP.
- This statement is recommended for system testing only.

POSTMASTER Statement

Use the POSTMASTER statement to specify the user ID to which SMTP will deliver all mail addressed to POSTMASTER.

Syntax



Parameters

user_id
The user ID on the local system to which mail addressed to POSTMASTER should be delivered. The default ID is TCPMAINT.

user_id@node_id
The NJE or SMTP address to which mail addressed to POSTMASTER should be delivered.

Examples

Set the user ID to receive the POSTMASTER mail to MAILGUY at POSTOFC:

```
POSTMASTER MAILGUY@POSTOFC
```

Usage Notes

- To specify multiple recipients to receive mail addressed to POSTMASTER, code a separate POSTMASTER statement for each recipient. There is no limit on the number of POSTMASTER statements that you can coded.
- In SECURE mode, you can only specify the POSTMASTER statement once for a local user ID as the single recipient of mail addressed to POSTMASTER.

RCPTRESPONSEDELAY Statement

Use the RCPTRESPONSEDELAY statement to specify how long the SMTP server delays responding to the RCPT commands from the sender SMTP, while it is waiting for domain name resolution.

Syntax

Parameters

seconds
A number in the range of 0 through 86400 specifying the number of seconds SMTP waits before responding to the RCPT TO command. The default is 60 seconds.

Examples

Set the RCPT TO: response time to 90 seconds:

```
RCPTRESPONSEDELAY 90
```

Usage Notes

If resolution does not complete before the specified period, the SMTP server assumes name resolution is successful and does the following:

- Sends the following message to the sender SMTP: `250 ok`.
- Queues the recipient address for asynchronous resolution.

If SMTP later determines that the recipient address cannot be resolved, the mail is returned to the sender.

RESOLVERRETRYINT Statement

Use the RESOLVERRETRYINT statement to specify the number of minutes SMTP waits between attempts to resolve domain names.

Syntax

Parameters

minutes
A number in the range of 1 through 1439 specifying the number of minutes between each attempt to resolve a domain name if the name server is causing delays. The default is to retry resolution every 20 minutes.

Examples

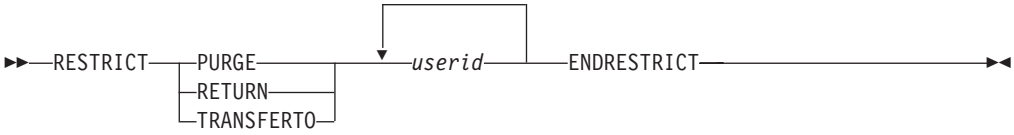
Set the waiting time between attempts to resolve domain names to 30 minutes.

```
RESOLVERRETRYINT 30
```

RESTRICT Statement

Use the RESTRICT statement to specify addresses of users who cannot utilize SMTP services.

Syntax



Parameters

PURGE
Specifies that the spool data set is to be purged.

RETURN
Specifies that the spool data set is to be returned to the originator.

TRANSFERTO
Specifies that the spool data set is to be forwarded to the specified *userid*.

userid
Specifies the address of the user.

Examples

In this example, mail from restricted users is returned, no mail is accepted from KNIGHT at 3 different nodes, and no mail is accepted from anyone on the host CASTLE.

```
RESTRICT RETURN
  KNIGHT@CAMPTENT
  KNIGHT@TOURNMNT
  *@CASTLE
ENDRESTRICT
```

Usage Notes

- The ENDRESTRICT statement ends the RESTRICT statement.
- If SMTP receives a spool data set from a restricted user, the spool data set is:
  - Purged, if PURGE is specified
  - Returned to the originator, if RETURN is specified
  - Forwarded to a specific user ID, if TRANSFERTO is specified

  In addition, SMTP rejects any MAIL FROM or RCPT TO commands whose destinations are restricted users.
- The TCPIP and NJE address must be included in the RESTRICT statement list in order to restrict a user from sending and receiving mail. 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.
- When the RESTRICT statement is used, incoming mail must be in NETDATA format.

- The RESTRICT statement cannot be used if the SMTP server is running as a secure gateway. Either remove or comment out the RESTRICT statements from the SMTP configuration data set.
- The RESTRICT statement cannot be used in combination with the SECURE statement.

Related Topics

- “LOCALFORMAT Statement” on page 671
- “MAILER Statement” on page 673
- “NJEFORMAT Statement” on page 681
- “SECURE Statement” on page 694

RETRYAGE Statement

Use the RETRYAGE statement to specify the number of days after which SMTP returns mail as undeliverable. SMTP tries to deliver mail to an inactive site. After the number of days specified on this statement, SMTP returns the mail to the sender with a note listing any recipients to which the mail could not be delivered.

Syntax

Parameters

days
A number in the range of 1 through 365 specifying the number of days to try to deliver the mail. The default is for SMTP to try to deliver a piece of mail for 3 days before returning it.

Examples

Keep trying to deliver mail for 2 days:

```
RETRYAGE 2
```

Related Topics

"WARNINGAGE Statement" on page 699

RETRYINT Statement

Use the RETRYINT statement to specify the number of minutes SMTP should wait between attempts to deliver mail to an inactive host.

Syntax

Parameters

minutes
A number in the range of 1 through 1439 specifying the number of minutes between each attempt to deliver the mail. The default is to try to establish a connection to these sites every 20 minutes.

Examples

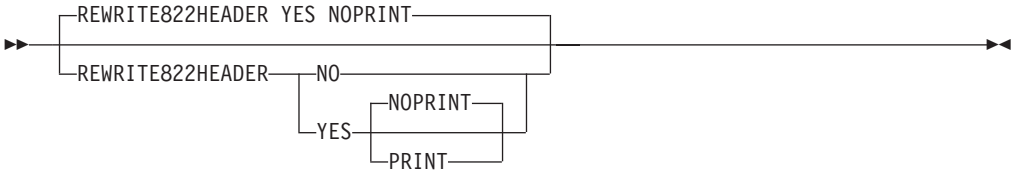
Try to re-deliver mail every 30 minutes:

```
RETRYINT 30
```

REWRITE822HEADER Statement

Use the REWRITE822HEADER statement to specify whether SMTP should rewrite or print the RFC 822 headers of mail arriving from the NJE side of the mail gateway.

Syntax



Parameters

NO
Specifies 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.

NOPRINT
Specifies that SMTP should not print the RFC 822 header rewriting rules to the console when SMTP starts.

PRINT
Specifies that SMTP should print the RFC 822 header rewriting rules to the console when SMTP starts.

YES
Specifies 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.

Examples

Rewrite the RFC 822 headers on all mail passing from NJE to TCP through the mail gateway and print the rules to the SMTP output when SMTP starts.

```
REWRITE822HEADER YES PRINT
```

Usage Notes

The SMTP.RULES data set specifies how the server is to rewrite the headers.

Related Topics

- “Step 4: Customize the SMTP Mail Headers (Optional)” on page 638
- “GATEWAY Statement” on page 665

SECURE Statement

Use the SECURE statement to specify that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.

Syntax

Parameters

None.

Usage Notes

- The SECURE statement cannot be used in combination with the RESTRICT statement.
- Mail will be accepted through the secure gateway only if the NJE user IDs and node IDs are included in the SMTP security table (SMTP security data set).
- When the SECURE statement is used, mail must be in NETDATA format.
- If you specify the SECURE statement, then source routing will be disabled to prevent the gateway from relaying mail to unauthorized users.

The data set pointed to by the //SECMEMO DD statement in the SECTABLE data set will be sent to NJE users that are not authorized to use the gateway.

Related Topics

- “Step 7: Create an SMTP Security Table (Optional)” on page 653
- “LOCALFORMAT Statement” on page 671
- “MAILER Statement” on page 673
- “NJEFORMAT Statement” on page 681
- “RESTRICT Statement” on page 689

SMSGAUTHLIST Statement

Use the SMSGAUTHLIST statement to specify the local users authorized to issue privileged SMTP SMSG commands. Any TSO 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 of SMTP and the enabling or disabling of various SMTP trace and debug options.

Syntax

Parameters

user_id
Specifies the address of a local user ID authorized to issue privileged SMTP SMSG commands. The *user_id* parameter can be repeated.

Examples

Specify the local users authorized to issue privileged SMTP SMSG commands:

```
SMSGAUTHLIST
    TCPMAINT
    OPERATOR CHANCE
ENDSMSGAUTHLIST
```

Usage Notes

The ENDSMSGAUTHLIST statement ends the SMSGAUTHLIST statement.

Related Topics

OS/390 SecureWay Communications Server: IP User's Guide

SPOOLPOLLINTERVAL Statement

Use the SPOOLPOLLINTERVAL statement to specify the interval (in seconds) for SMTP to check the spool for incoming batch data sets.

Syntax

Parameters

seconds
The number of seconds between each check. The range is from 5 to 3600 seconds (3600 seconds equals one hour).

Examples

Set the time between spool polling to 30 seconds:

```
SPOOLPOLLINTERVAL 30
```

Usage Notes

If the value for *seconds* is too low, system overhead is increased; if the value is too high, incoming mail must wait to be processed.

TEMPERRORRETRIES Statement

Use the TEMPERRORRETRIES statement to specify 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.

Syntax

Parameters

retries
The number of times mail delivery to a host with a temporary problem is retried. The default is 0.

Examples

Try to re-deliver 5 times when there is a temporary problem with the host to which mail is addressed:

```
TEMPERRORRETRIES 5
```

Usage Notes

- If delivery is still unsuccessful, the mail is returned to the sender.
- Change the number of *retries* from the default of 0 only when remote mail servers repeatedly terminate abnormally or hang SMTP mail transactions.
- If *retries* is 0 and there is a problem with the remote mail server, then SMTP will continue re-trying to deliver the same piece of mail until it times out. The other mail sitting behind it in the queue will wait for delivery until SMTP times out.

TIMEZONE Statement

Use the TIMEZONE statement to specify the printable name of the local time zone.

Syntax

Parameters

time_zone
The name of the local time zone. This parameter must be exactly 3 characters long. The default name is LCL.

Examples

Set the time zone to pacific standard time (PST):

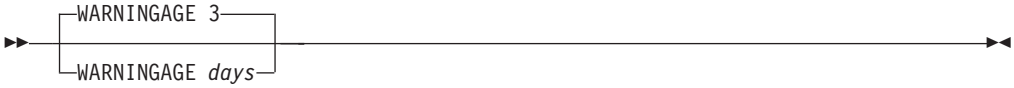
```
TIMEZONE PST
```

WARNINGAGE Statement

Use the WARNINGAGE statement to specify the number of days after which a copy of the mail is returned to the sender with a warning. The warning is included in the header in the copy of the mail, It says that:

- SMTP has been unable to deliver the mail so far
- How many days it has been undeliverable
- How many days that SMTP will continue to try to deliver the mail (taken from the RETRYAGE statement)

Syntax



Parameters

days
A number from 0 through 365 specifying the number of days to attempt delivery of the mail before sending a nondelivery warning to the sender. The default is 3 (the same as the default for the RETRYAGE statement).

Examples

Warn the sender that mail has been undeliverable for one day, but that SMTP will try to deliver the mail for another two days:

```
RETRYAGE 3
WARNINGAGE 1
```

Usage Notes

SMTP will only send a warning if the number of days specified on the WARNINGAGE statement is less than the number of day specified on the RETRYAGE statement. When the number of days specified by the WARNINGAGE statement is more than or equal to the number of days specified on the RETRYAGE statement, then no warning is issued to the sender.

Related Topics

"RETRYAGE Statement" on page 691

Operating the SMTP Server

This section gives a brief overview of the SMSG interface to SMTP and explains how to force the re-resolution of queued mail.

Using SMSG with SMTP

The SMSG command provides an interface to the SMTP address space. Using the SMSG SMTP command you can query the operating statistics or the mail delivery queues of the SMTP server with the following options:

- HElp
- QUeues
- STats

You can also do privileged system administration tasks with the following options:

- SHutdown
- TRace
- NOTrace
- DEbug
- NODebug
- EXpire

For more detailed information about the SMSG SMTP command, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

Forcing Re-resolution of Queued Mail

Normally, the SMTP server resolves the MX or A records of a piece of mail and stores the mail in the data sets pointed to by the MAILFILEDSPREFIX keyword in the SMTP configuration data set. If the mail cannot be delivered for some period of time, the IP addresses in the mail can become old or obsolete. The data set names for each piece of mail are:

mailfiledsprefix.number.ADDRBLOK
mailfiledsprefix.number.NOTE

To force the SMTP server to reresolve the addresses, modify the ADDRBLOK data set for the piece of mail. For each recipient record (records 3 through the end of the data set), if the first character of the record is an `S`, then change the `S` to an `E`, for expired. This causes SMTP to reresolve that record in the ADDRBLOK data set the next time the SMTP server is started.

To modify the ADDRBLOK data set, the data set must be zapped, or a local utility program must be used. The data set cannot be modified using the ISPF editor or IEBUPDATE.

Chapter 20. Configuring OS/390 UNIX sendmail and Popper

Before You Configure...

This chapter is intended to provide the administrator with specific information on how to configure sendmail on the OS/390 platform. Before using this chapter, become familiar with the industry-accepted publication for sendmail, *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). That publication is known throughout the industry as simply the *"bat book"*, and this chapter consistently refers to the *"bat book"* for further information.

Additional information about sendmail can also be found on the OS/390 UNIX application website, http://www.s390.ibm.com/unix, as well as in documents from the sample directory that were received during the port of sendmail 8.8.7 from the http://www.sendmail.org website. The Sendmail Installation and Operation Guide document (sendmail.ps), for instance, is the generic guide from http://www.sendmail.org, which might be helpful as a more thorough guide in a slightly different format. The README.m4 document gives more details for building a configuration file using the m4 preprocessor.

This chapter also provides information on how to configure popper on the OS/390 platform. The popper function requires very little configuration. For more information on this UNIX application, see RFC1939.

Overview

The simple mail architecture in which sendmail and popper fit includes a mail user agent (MUA), a mail transfer agent (MTA), and a mail delivery agent (MDA). An MUA is client software that a user invokes directly to send and receive email. Examples of MUAs include Eudora, Netscape Navigator, pine and elm. An MTA is software that actually routes messages from a sender's system to the receiver's system. Sendmail is an MTA. It's worth noting, however, that sendmail relies on other programs to implement non-SMTP based transport (for example, UUCP-based transport as well as local delivery to a user's mail spool file). An MDA is server software that delivers received mail to a user's MUA. Popper is an example of an MDA using the POP3 protocol.

At the sender's end of the mail delivery process, the sender's MUA transmits the message to be delivered to sendmail, as shown in Figure 26.

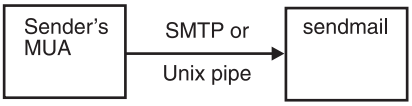


Figure 26. Sender MUA Transmits the Message to sendmail

This can occur in one of two ways. If the MUA is running on the local host, the message can be transmitted by executing a copy of sendmail and transmitting the message to the standard input of that process via a Unix pipe.

Alternatively (and more commonly), a copy of sendmail will be running as a daemon, and the MUA (running on either the local host, or on a remote host) will open an SMTP connection to the sendmail daemon, transmitting the message to be

delivered via that SMTP connection. In this case, sendmail is acting as an SMTP server, while the MUA is acting as an SMTP client.

In the next step, for each recipient address, sendmail transmits the message to some other SMTP server, to route the message to its final destination at the recipient's site. This is shown in Figure 27.

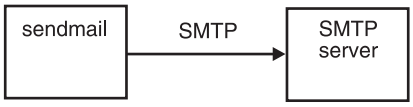


Figure 27. sendmail Transmits the Message to an Intermediate SMTP Server

The receiving SMTP server, in this case, might be a local hub that handles all mail at the sender's site, a remote hub handling all mail at the recipient's site, or an SMTP server at the recipient's host system.

In the next step, sendmail acts as an SMTP client, initiating an SMTP connection with some SMTP server, and then transmitting the message to be delivered to that server, via the SMTP connection.

At the receiver's end of the mail delivery process, a sendmail daemon receives the message from some SMTP client, as shown in Figure 28.

Figure 28. A sendmail Daemon Receives the Message from an SMTP Client

The sendmail daemon, acting as an SMTP server, accepts an incoming SMTP connection, and receives a message to be delivered over that SMTP connection. (This is identical to receipt of a message from an MUA, over an SMTP connection.)

Upon receiving the message, sendmail delivers it to the local recipient by appending the message to the recipient's mail spool file. To do this, sendmail requires a local mailer program, as depicted in Figure 29.

Figure 29. Sendmail Delivers the Message to the Local Recipient

In this step, sendmail executes a specified local mailer program, such as /bin/mail, and transmits the message to be delivered to that mailer via a Unix pipe. The mailer program appends the message to the recipient's mail spool file. With this sendmail's role in delivery of mail is completed.

For the recipient to now read the received message, an MUA must be used. As mentioned in the previous section, depending upon the MUA, this may or may not require an additional MDA, such as popper. If the receiver's MUA has direct access

to the mail spool file, the MUA may retrieve the mail directly from the spool file, as depicted in Figure 30.

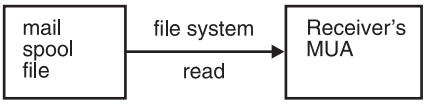


Figure 30. Receiver's MUA has Direct Access to the Mail Spool File

Alternatively (and more commonly), the MUA will establish a POP3 connection with a popper daemon, and retrieve the message over that connection. This is shown in Figure 31.

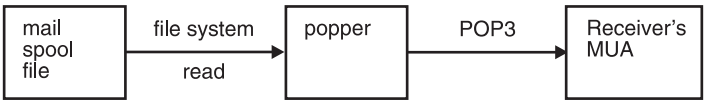


Figure 31. Receiver's MUA Retrieves the Message over a POP3 Connection with a Popper Daemon

The popper daemon will also allow the receiver's MUA to manage the mail spool file, by allowing it to specify whether and which message should be deleted.

Configuring OS/390 UNIX sendmail

This section contains information on the following:

- The sendmail samples directory
- Creating the configuration file
- Creating an aliases file
- Configuration hints and tips

The sendmail Samples Directory

Much of the sendmail samples directory is dedicated to the automated creation of the configuration file. The /usr/lpp/tcpip/samples/sendmail/cf directory contains a sample.mc file and the subsequent sample.cf configuration file that was created by running the m4 macro preprocessor on the sample.mc file. If the /usr/lpp/tcpip/sample/sendmail directory is examined, the following directory structure can be found:

```
cd /usr/lpp/tcpip/sample/sendmail
ls
README.m4    feature   mailer      siteconfig
cf           hack      ostype      sendmail.ps
domain       m4        sh
```

cf

Both site-dependent and site-independent descriptions of hosts. Files ending in **.mc** (″Master Configuration″) are the input descriptions. The output is in the corresponding **.cf** file. The general structure of these files is described below.

domain

Site-dependent subdomain descriptions. These are tied to the way your

organization wants to do addressing. These descriptions are referenced using the DOMAIN *m4* macro in the **.mc** file.

feature
Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE *m4* macro. An example feature is `use_cw_file`, which tells OS/390 UNIX sendmail to read an `/etc/sendmail.cw` file on startup to find the set of local names.

hack
Local hacks, referenced using the HACK *m4* macro. Avoid these.

m4
Site-independent *m4(1)* include files that have information common to all configuration files. Think of this as a ″#include″ directory.

mailer
Definitions of mailers, referenced using the MAILER *m4* macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use ″MAILER(uucp)″.

ostype
Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE *m4* macro. This directory contains only the os390 definition.

README
Contains all the latest information regarding this latest version of sendmail from the *www.sendmail.org* site

sh
Shell files used by the *m4* build process.

siteconfig
Local UUCP connectivity information. These normally contain lists of site information, for example:

- SITE(contessa)
- SITE(hoptoad)
- SITE(nkainc)
- SITE(well)

These are referenced using the SITECONFIG macro:

```
SITECONFIG(site.config.file, name_of_site,X)
```

where *X* is the macro/class name to use. It can be U (indicating locally connected hosts) or one of W, X, or Y for up to three remote UUCP hubs. This directory has been supplanted by the mailer table feature. Any new configurations should use that feature to do UUCP (and other) routing.

sendmail.ps
This is a PS file of the *Sendmail Installation and Operation Guide* provided by *www.sendmail.org* in this version of sendmail.

Creating the Configuration File

The basic steps to create the configuration file are:

1. Retrieve the m4 preprocessor.
2. Create the *.mc* file.

3. Build the configuration file.

Retrieving the m4 Preprocessor

Retrieve the m4 macro preprocessor from the OS/390 Toys and Tools webpage at: `http://www.s390.ibm.com/unix/bpxa1toy.html`

The m4 macro preprocessor can be given input that will generate an OS/390 UNIX sendmail configuration file. It takes as input a user-defined master configuration source file (.mc file) that can define mail delivery mechanisms using files provided in the samples directory. For more information on the .mc file, see “Creating the .mc File”.

The m4 preprocessor is downloaded as m4_pax.Z. To *unpax* the file, issue the following command:

```
pax -rf m4.pax.Z -o from=iso8859-1,to=ibm-1047
```

Creating the .mc File

The process of building an OS/390 UNIX sendmail configuration file begins by creating a file of m4 statements. The suffix for this file is *.mc*.

The Minimal mc File: Every **.mc** file must contain minimal information. This file defines the mail delivery mechanisms understood at this site, how to access them, how to forward email to remote mail systems, and a number of tuning parameters. The following table shows which items are required and also which items are recommended. It is recommended that the starting point for these items be as shown in the sample.mc file, and an investigation of all the m4 techniques that are available to customize the .mc file for your mail server is encouraged [refer to *sendmail* by O’Reilly & Associates, Inc. (ISBN 1-56592-222-0); this book is also referred to as the *bat book*].

Table 25. Required and Recommended m4 Items

Item	*"Bat Book"* **Reference**	**Required or Recommended**	**Description**
OSTYPE()	19.3.1	Required	Support for your operating system
MAILER()	19.3.2	Required	Necessary delivery agent
DOMAIN()	19.3.3	Recommended	Common domain wide information
FEATURE()	19.3.4	Recommended	Solutions to special needs

Example files can be found in the `/usr/lpp/tcpip/samples/sendmail` directory. The *cf* directory contains an example of an **.mc** file. Of special interest are the files that begin with *generic*. These can serve as template statements in developing customized **.mc** files. The following is an example of a simple **.mc** file.

```
divert (-1)
divert(0) dnl
VERSIONID('OS/390 sample configuration 12/4/97')
OSTYPE(os390)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

Following is a description of these common *m4* items. For more information on these items, refer to the *"bat book".*

divert

- (-1) Ignore the lines following
- (0) Stop diverting and output immediately

VERSIONID
Used to insert an identifier into each **.mc** and **.m4** file that will become your header.

OSTYPE()

- Support for operating system (the only ostype provided in the `/usr/lpp/tcpip/samples/sendmail/ostype` directory is **os390.m4**
- Required

MAILER()

- Necessary delivery agent
- Required
- Known values include:
  - fax
  - local
  - smtp
  - uucp
  - usenet

DOMAIN()
Common domain wide information

FEATURE()
Solution to special needs

Building the Configuration File

To build the configuration file, go to the directory containing the m4 executable and issue the following command:

```
m4  ../m4/cf.m4 yourmcfile.mc > yourcffile.cf
```

where *yourmcfile* is the name of your .mc file and *yourcffile* is the name you want to give your .cf file.

The `../m4/cf.m4` specifies the master prototype configuration file `cf.m4` in the `m4` directory of the samples/sendmail directory. This is the path to the samples/sendmail directory structure from the location of your m4 executable. This can also be specified in your .mc file using *include* as follows:

```
include('../m4/cf.m4')
```

Creating the Aliases File

Aliasing is the process of converting one recipient name into another; a generic name (such as root) into a real username, or one name into a list of names (that is, a mailing list). Define the location of your aliases file using the AliasFile option in your sendmail.cf file. For example:

```
AliasFile=/etc/aliases
```

For sendmail to work, aliases are required for MAILER-DAEMON and postmaster. Every aliases file must include these required aliases.

The alias for postmaster must expand to the name of a real user, based on the requirement that every site has to be able to accept mail addressed to a user named postmaster. Unless a site has real user account named postmaster, an alias is required in the aliases file. The postmaster receives mail about mail problems sent by mail-related programs and by users that are having trouble sending mail.

When mail is bounced (returned because it could not be delivered), it is sent from MAILER-DAEMON but it is shown as being the original sender who sent the mail. This alias is defined because users often inadvertently reply to the bounced mail.

Following is an example of an aliases file. Lines that begin with # are comments. Empty lines are ignored. For more information on the different forms of aliases, refer to the *bat book.*

```
# Alias for mailer daemon
MAILER-DAEMON:IBMUSER

# Following alias is required by the new mail protocol, RFC 822
postmaster:IBMUSER

# Alias to handle mail to msgs and news
nobody: /dev/null
```

After the aliases file is created and before the sendmail daemon is brought up for the first time, the aliases file must be loaded by running sendmail using the *newaliases* command or with the *-bi* command-line switch.

For more information on the aliases file, refer to the *bat book.*

Configuration Hints and Tips

This section contains other required or useful information for configuring sendmail. For further information on these topics, refer to the *"bat book".*

- SuperUser status is needed to start the sendmail daemon.
- The QueueDirectory option defined in the config file tells sendmail where to queue messages that are temporarily undeliverable. This directory must exist before sendmail is started.
- Sendmail is highly dependent on the Domain Name Server (DNS); it is important that the resolver be set up correctly to avoid unnecessary searching for a user. For more information on DNS, see "Chapter 21. Configuring the Bind-Based Domain Name System (DNS)" on page 709.
- Table 26 shows the expected file permissions of files that sendmail might use.

Table 26. Sendmail Permission Table

Path	**Type**	**Owner**	**Mode**		*"Bat Book"* **Reference**
/	Directory	*root*	0755	drwxr-xr-x	22.1
/usr	Directory	*root*	0755	drwxr-xr-x	18.8.34
/usr/sbin/sendmail	File	*root*	06511	-r-s--s--x	Entire Book
/etc	Directory	*root*	0755	drwxr-xr-x	18.8.34
/etc/sendmail.cf	File	*root*	0644 or 0640		Chapter 27
/etc/sendmail.st	File	*root*	0644	-rw-r--r--	26.6
/etc/sendmail.hf	File	*root*	0444	-r--r--r--	34.8.28
/etc/aliases	File	*root*	0644	-rw-r--r--	Chapter 24

Table 26. Sendmail Permission Table (continued)

Path	Type	Owner	Mode	"Bat Book" Reference
/etc/aliases.pag	File	*root*	0644 -rw-r--r--	24.5
/etc/aliases.dir	File	*root*	0644 -rw-r--r--	24.5
/etc/aliases.db	File	*root*	0644 -rw-r--r--	24.5

If a system has thousands of users defined in the Users list, the administrator might consider enabling the UNIXMAP class. This increases the speed of the security checks performed by sendmail. APAR OW30858 provides details about what is needed to enable the UNIXMAP class.

For additional information about enabling the UNIX map class, refer to *OS/390 Security Server (RACF) System Planning: Installation and Migration*.

Configuring Popper

Popper must be invoked by INETD upon the initiation of a TCP connection to the POP3 port 110 (or any other specifically-configured port defined in /etc/services - port 110 is the well-known port for POP3 protocols). Therefore, you must add the following command lines to your /etc/inetd.conf file:

```
pop3 stream tcp nowait bpxroot /usr/sbin/popper popper -d
```

The above must be added to your /etc/inetd.conf file and INETD must be started with this configuration file. For more information on inetd, see “Appendix F. Setting up the inetd Configuration File” on page 1191.

POP3 resides on port 110. You can define additional ports if there is a need for additional command-line options for popper. For information on the options that might be suitable for your site, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

OS/390 UNIX popper will most likely be used by those whose local mailer requires a POP3 server. Typically their administrator will provide them with the address or name of the OS/390 running the POP3 server, with instructions on where this information should be used.

Chapter 21. Configuring the Bind-Based Domain Name System (DNS)

This chapter contains information about configuring the name server in a BIND-based Domain Name System (DNS). BIND (Berkeley Internet Name Domain) is the most common implementation of a DNS. BIND was developed at the University of California, Berkeley, and is currently maintained by the Internet Software Consortium (ISC). The name server is based on BIND 4.9.3.

This chapter also contains information about connection optimization which uses DNS for distributing connections among hosts or server applications within a sysplex domain.

The Domain Name System is a client/server model in which programs called *name servers* contain information about host systems and IP addresses. Name servers provide this information to clients called *resolvers*.

This chapter is not intended to be a comprehensive description of DNS or of BIND. For more complete descriptions, refer to texts such as *DNS and BIND, 2nd Edition* by Paul Albitz and Cricket Liu (O'Reilly & Associates, Inc., 1997).

DNS and BIND Overview

While TCP/IP applications refer to host computers by their IP addresses, human beings find it easier to use host names. To enable the use of host names in a network, the Domain Name System translates host names to IP addresses. DNS provides the host name-to-IP address mapping through network server hosts called *domain name servers*. (For detailed information about name servers, see "Domain Name Servers" on page 710.) DNS can also provide other information about server hosts and networks such as the TCP/IP services available at a server host and the location of domain name servers in a network.

DNS organizes the hosts in a network into domains. A *domain* is a group of hosts that share the same name space in the domain hierarchy and are usually controlled within the same organization. Domains are arranged in a hierarchy. A special domain known as the *root domain* exists at the top of the hierarchy. The root domain servers store information about server hosts in the root domain and the name servers in the delegated, *top-level* domains, such as *com* (commercial), *edu* (education), and *mil* (military). The name servers in the top-level domain, in turn, store the names of name servers for their delegated domains, and so on.

The complete name of a host, also known as the *fully qualified domain name* (FQDN), is a series of labels separated by dots or periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to one of the larger networks generally has more than one subdomain, as shown in these examples:

```
host1.subdomain2a.subdomain2.rootdomain
user4720.eng.mit.edu
```

Note: The dots or periods are separators only and should not be part of a label. For example, host1 or host-1 are valid labels and host.1 is not.

A domain name server requires the FQDN. The resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

DNS also provides IP address-to-host name mapping using a special domain called *in-addr.arpa*. This kind of mapping is useful for producing output (host names) that is easy to read. The format of an in-addr.arpa name is the reverse octet of an IP address concatenated with in-addr.arpa. For example, the address 9.67.30.143 has an in-addr.arpa name of 143.30.67.9.in-addr.arpa.

As system administrator, you can name the host systems and domains in your local, private network anything you want, but to link with name servers in a public network like the Internet, you need to determine which domain you want to be in (which parent domain) and then contact the registrar in that domain to get the names and IP addresses of your name servers registered. This ensures that your local database becomes part of the DNS distributed database.

Note: Contact the InterNetwork Information Center (InterNIC) for more information about Internet registration. You can contact InterNIC by pointing your Web browser at `http://ds.internic.net`.

Domain Name Servers

A name server is said to be *authoritative* for some part of the domain name space, called a *zone*. A zone consists of the resources within a single domain (for example, commercial or `.com`) or subdomain (for example, `raleigh.ibm.com`). Typically, a zone is administered by a single organization or individual.

All host systems in a given zone share the same higher level domain name (for example, `host1.raleigh.ibm.com`, `host2.raleigh.ibm.com`, `host3.raleigh.ibm.com`, and so on). As system administrator, you create a zone of authority by listing all the host systems in your zone in the database file of the name server that is authoritative for the zone.

If a domain name server receives a query about a host for which it has information in its database or in its cache, it performs the name resolution and returns all the address records associated with the host to the client. Some hosts (for example, routers or gateways between two or more networks) may have more than one IP address.

Alternatively, the name server can ask other name servers for information. This process is called *iterative resolution*. The local name server successively queries other name servers, each of which responds by referring the local name server to a remote name server that is closer to the name server authoritative for the target domain. Finally, the local name server queries the authoritative name server and gets an answer. If the information about a requested host name does not exist or if a name server does not know where to go for the information, it sends a negative response back to the client.

There are four kinds of name servers in the DNS:

- Master servers (primary and secondary)
- Caching-only servers
- Forwarders
- Slaves

A single server can perform multiple functions—for example, it can be a primary server and a secondary server for different zones. The purpose of having these different kinds of servers is to provide redundancy (in case of system failure), to distribute the workload among multiple servers, to speed up the name-resolution process, and to provide flexibility in network design.

Master servers, caching-only servers, forwarders, and slaves are discussed in the following sections.

Master Servers

A master server is the authority for its domain. It queries and is queried by other name servers in the DNS. The data it receives in response from other name servers is cached. Master servers do not consider themselves authoritative for cached data.

There are two types of master servers: primary and secondary. Each domain must have one, and only one, primary name server, and it should have at least one secondary name server for backup.

Primary Name Servers: A primary name server maintains all the data for its zone. Static resources are kept in database files called *domain data files*. For information on creating domain data files, see “Step 4. Create the Domain Data Files (Primary Name Server Only)” on page 715. Primary name servers can also receive zone updates dynamically. For information on dynamic DNS, see “Dynamic IP” on page 737. For information on dynamic generation of resources, see “Connection Optimization in a Sysplex Domain” on page 723.

Secondary Name Servers: A secondary name server acts as an alternate to the primary server if the primary name server becomes unavailable or overloaded. The secondary name server receives zone data directly from the primary name server in a process called *zone transfer*. Zone transfers, which only occur when data has changed, are based on the refresh interval in the Start of Authority (SOA) resource record. (For a description of the SOA resource record, see “Resource Records” on page 787.) A secondary server, like a primary server, is authoritative for a domain.

Caching-Only Servers

All name servers cache (store) the data they receive in response to a query. A caching-only server, however, is not authoritative for any domain. When a caching-only server receives a query, it checks its cache for the requested information. If it does not have the information, it queries a local name server or a root name server, passes the information to the client, and caches the answer for future queries. The names and addresses of root name servers are stored in its hints (root server) file, the name and filepath of which are specified in the name server’s boot file.

You can use caching servers to create a large cache of responses to frequently requested queries and reduce the number of queries made to master servers. The caching server stores data for a period of time determined by the time-to-live (ttl) value, and the cached information is lost if the name server is restarted.

Forwarders

Normally, name servers answer queries from cached data or, if that does not succeed, they attempt to contact other name servers identified in their data files as

authoritative for certain domains. However, name servers can also be configured to contact special servers called *forwarders* before contacting the name servers listed in their data files. If a forwarder cannot process the query and if the local name server is not a slave (see definition below), the local name server contacts the name servers in its data files.

The forwarding function is useful for reducing the number of queries to servers on the Internet and for creating a large cache of information on forwarders. It is also a useful function for providing Internet access for local servers that, for one reason or another, do not have access themselves.

Slaves

Slave servers answer queries with their own authoritative data or cached data. However, they can only send queries to forwarders, which serve as links to other networks. A slave server does not attempt to contact name servers that are not forwarders.

Note that a name server can use a forwarder and not be a slave. Slave name servers, however, must use forwarders.

Recommended Reading

In addition to books like *DNS and BIND*, you can also read the *Name Server Operations Guide for BIND* available from the ISC. To obtain copies of the guide, point your browser to `http://www.isc.org/isc/bind.html`.

For information on Dynamic IP, see “Dynamic IP” on page 737.

If you wish to request participation in a mail group that discusses issues related to BIND, contact `bind-request@uunet.uu.net`.

DNS protocols are described in various Request for Comments (RFC) papers and Internet drafts. RFCs outline existing protocols, suggest new protocols, and establish standards for the Internet protocol suite. Internet drafts are proposals, techniques, and mechanisms that document Internet Engineering Task Force (IETF) work-in-progress.

Online copies of RFCs and Internet drafts are available from the InterNIC. To access the Internet Documentation and IETF Information home page, point your Web browser at `http://ds.internic.net/ds/dspg0intdoc.html`. To transfer RFCs and Internet drafts, point your browser at `ftp://ftp.ds.internic.net`. Alternatively, you can use FTP to connect to `ds.internic.net`, then transfer the files from the RFC directory using the following format:

```
RFCn.TXT
RFCn.PS
```

where:

```
n      is the RFC number
TXT    indicates text format
PS     indicates PostScript format
```

In the RFC directory, the following is the format for the RFC index:

```
RFC–INDEX.TXT
```

Note: Many RFCs are available only in text format. Before requesting a PostScript file, first check the RFC index to make sure the RFC is available in that format.

You can also request online copies of RFCs through electronic mail, from the automated InterNIC mail server, by sending a message to `mailserv@ds.internic.net`. You must include one of the following commands in the body of your note:

```
document-by-name  RFCn.TXT
document-by-name  RFCn.PS
```

where:

```
n      is the RFC number
TXT    indicates text format
PS     indicates PostScript format
```

For example, to request the text format of RFC 1034, specify the following command in your note:

```
document-by-name   RFC1034.TXT
```

To request an online copy of the RFC index, specify the following command in your note:

```
SEND   RFC-INDEX.TXT
```

The following three RFCs contain basic information about the DNS:

1033 *Domain Administrators Operations Guide*, M. Lottor

1034 *Domain Names—Concepts and Facilities*, P.V. Mockapetris

1035 *Domain Names—Implementation and Specification*, P.V. Mockapetris

Setting Up and Running the Name Server

This section describes the three tasks involved in setting and running the IBM Domain Name System:

- Migrating to the name server
- Configuring the name server
- Configuring host resolvers (name server considerations)
- Creating the syslog file

Name server configuration files are arranged in a Hierarchical File System (HFS). Before configuring DNS, the TSO userid from which the name server is started must have the proper authority to access the name server boot and zone files. For a complete description of file permissions within the HFS, refer to *OS/390 UNIX System Services Planning*.

Migrating to the Name Server

Copy the zone files used for the DB/2-based name server to the HFS for the BIND-based name server using the OPUT command. For a complete description of this command, see *OS/390 UNIX System Services Command Reference*.

Configuring the Name Server

The name resolution process is an example of a client/server relationship in which clients, through their resolvers, request a service (name resolution) from name servers. (For a general overview of name servers, see “Domain Name Servers” on page 710.) The steps for configuring a master server or a caching-only server are summarized below:

1. Specify stack affinity (multiple stack environment).
2. Specify port ownership.
3. Update the name server start procedure.
4. Create the domain data files (primary name server only).
5. Create the hints (root server) file.
6. Create the loopback file.
7. Create the boot file.
8. Configure the start process.

The only difference between the configuration process for a primary name server and the configuration processes for secondary and caching-only servers is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). The domain data files are maintained on the primary name server, and the secondary name server transfers this data to its own database.

To configure a server to act as a slave, you use two boot file directives, `options forward-only` and `forwarders`. For descriptions of these directives and other boot file directives, see “Boot File Directives” on page 792.

Step 1. Specify Stack Affinity (Multiple Stack Environment)

In a multiple stack environment, the name server binds with the default stack. To bind the name server with a different stack, you must specify the job name using the TCPIPJOBNAME or TCPIPUSERID configuration parameters. For information about running coexistent stacks, see “Considerations for Multiple Instances of TCP/IP” on page 57.

Step 2. Specify Port Ownership

The name server uses a single port (53) for TCP and UDP sessions. To specify port ownership when using the named start procedure or when starting the name server directly from OS/390 UNIX, add the following statements to the *hlq*.PROFILE.TCPIP data set:

```
PORT
  53 TCP NAMED
  53 UDP NAMED
```

For more information on the PORT statement, see “PORT Statement” on page 229.

Step 3. Update the Name Server Start Procedure

Move the sample start procedure, `TCP.SEZAINST(NAMED)`, to your system or recognized PROCLIB. Specify name server parameters and change the data set names as required to suit your local configuration. You can also change the boot file path which, in the sample start procedure below, is `/etc`:

```
//NAMED PROC B='/etc/named.boot',P='53'
*****************************************************************
//NAMED     EXEC PGM=EZANSNMD,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON)/ -b &B -p &P'
//*STEPLIB  DD DISP=SHR,DSN=TCP.SEZALINK
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN     DD DUMMY
//SYSERR    DD SYSOUT=*
//SYSOUT    DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP   DD SYSOUT=*
```

Step 4. Create the Domain Data Files (Primary Name Server Only)

When setting up domain files, determine whether domains will be static (default), dynamic, or balanced. For information on dynamic domains, see “Dynamic IP” on page 737. The domain data files contain information about a domain, such as the IP addresses and names of the hosts in the domain for which the primary name server is authoritative. The *forward* domain data file contains entries that provide forward mapping (host names-to-IP addresses for each host system in the zone) as well as additional information about system resources. The *reverse* domain data file contains entries that provide reverse mapping (IP addresses-to-host names). You create a separate reverse domain data file for each network (or subnet) in a domain.

Note: The TSO userid from which the name server is started must have the proper authority to access the name server boot and zone files. For a complete description of file permissions within the HFS, see the *OS/390 UNIX System Services Planning* (SC28–1890–02).

You can give the domain data files any names you want, but for convenience in maintaining the named database, give them names of the form `named.extension`, where the extension identifies the type of file. This guide uses the extension `.rev` to specify the reverse domain data file, `.for` to specify the forward domain data file, and `.bak` to specify a backup file. The server daemon, named, creates a backup file when a zone transfer occurs if a file is specified in the boot file. The secondary name server can use the backup file on subsequent initializations.

Create domain data files using the following elements:

- “Control Entries” on page 787
- “Resource Records” on page 787
- “Special Characters” on page 791

To see what forward and reverse domain data files look like, refer to “Sample Forward Domain Data File” on page 794 and “Sample Reverse Domain Data File” on page 795.

Note: Data files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. (See *OS/390 UNIX System Services Command Reference* for more detailed information about this command.) Files read through a network connection (for example, secondary data files) are converted to IBM-1047 by the name server before they are written to the local file system.

FTP may also be used to convert the files to code page IBM-1047.

Step 5. Create the Hints (Root Server) File

The hints file contains the names and IP addresses of the authoritative root domain name servers. The root name servers contain the names of name servers in the top-level domains such as com, edu, and mil. The name server uses root server information when deciding which name server to contact when it receives a query for a host outside its zone of authority and it does not have the data in its cache.

Note: The hints file does not contain cached data nor does the name server provide other hosts with the information contained in the hints file. A slave server is the only type of name server that does not require a hints file.

To obtain a hints file, point your Web browser at `ftp://ftp.rs.internic.net` and retrieve the file `named.root` from the `domain` subdirectory. Update your hints file on a regular basis. (For an example of a hints file, see “Sample Hints (Root Server) File” on page 795.)

The `cache` directive in a boot file specifies the path and name of the hints file. This guide uses the extension `.ca` to specify the hints file.

Step 6. Create the Loopback File

The loopback file contains the loopback address. This is the address that a host uses to route queries to itself. The preferred loopback address is `127.0.0.1`, although you can configure additional loopback interfaces in the TCP/IP profile. DNS will bind to `127.0.0.1` in addition to the first loopback address configured in *hlq*.PROFILE.TCPIP.

To get the sysplex domain name, add the following PTR record to the loopback file in the loopback zone:

```
127.0.0.128.in-addr.arpa.   IN PTR  Sysplex_Domain_Name.
```

Sysplex_Domain_Name is the domain name of the sysplex (specified as cluster zone in the primary boot file). Do not forget to put a period (.) after the *Sysplex_Domain_Name*. Note that all of the sysplex name servers must be updated with this change.

This guide uses the extension `.lbk` to specify the loopback file.

Note: In addition to creating the loopback file, add an address resource record called *localhost* to the forward domain data file. This record supports proper two-way resolution. The “Sample Forward Domain Data File” on page 794 contains a localhost record.

You create the loopback file using the following elements:

- “Control Entries” on page 787
- “Resource Records” on page 787
- “Special Characters” on page 791

See also “Sample Loopback File” on page 797.

Step 7. Create the Boot File

The boot file is the main configuration file for a domain name server. The named daemon reads the boot file for information about how to set up the local name

server. The records in the boot file identify the type of name server, the zones over which it has authority, the location of data for setting up its name resolution database, and other configuration options. The default name of the boot file is `/etc/named.boot`. You can specify an alternate boot file using the `-b` named start option. For information about named options, see “The named Daemon” on page 799.

Note: The named daemon reads the boot file only when the named daemon starts or when it receives a SIGHUP signal. For a description of named signals, see “The named Daemon” on page 799.

Each type of name server (for example, primary) has a special boot file configuration. You create a boot file using directives. (See “Boot File Directives” on page 792 and “Sample Boot File” on page 797.) A sample boot file is included with the product. Refer to the program directory for its location.

Boot files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. (See *OS/390 UNIX System Services Command Reference* for more information about this command.)

Step 8. Configure the Start Process

Use the following methods for starting the name server:

- If you are a supervisor with an authorized TSO ID, you can start a name server from the MVS operators console by starting the named start procedure. If the boot file path is not `/etc`, you can specify the correct path in the start procedure. (See “Step 3. Update the Name Server Start Procedure” on page 714.) A sample start procedure is provided with the product and is found in TCP.SEZAINST(NAMED).
- If you have superuser authority, you can start the name server from the shell by starting OS/390 UNIX, then issuing the `named` command and, optionally, any parameters. See “The named Daemon” on page 799 for a complete description of the named daemon.
- You can also start the server automatically when OS/390 UNIX is started by specifying the path and filename of the OS/390 UNIX initialization shell script in the `/etc/init.options` file using the `-sc` option:

  ```
  -sc /etc/rc       shell script = /etc/rc
  ```

  The file `/etc/rc` is the default OS/390 UNIX initialization shell script that is executed when OS/390 UNIX is started. (See the *OS/390 SecureWay Communications Server: IP Migration*.) Information such as the following may be entered in `/etc/rc`:

  ```
  # Start name server
  /usr/lpp/tcpip/sbin/named -b /named/production/named.boot &
  ```

  Port 53 must be reserved for NAMED in the *hlq*.PROFILE.TCPIP data set. (For directions on specifying port ownership, see “Step 2. Specify Port Ownership” on page 714.) When the stack to which named binds is started, named completes initialization.
- You can use the AUTOLOG statement to start the name server automatically during initialization with OS/390 UNIX running. Insert the name of the named start procedure in the AUTOLOG statement of the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  NAMED
ENDAUTOLOG
```

The jobname keyword should not be added to the AUTOLOG statement. For more information on the AUTOLOG statement, see “AUTOLOG Statement” on page 137.

Note: You cannot start named from `INETD`.

Configuring Host Resolvers: Name Server Considerations

If the name server will run on the host being configured, you need to create a loopback file. You also need to specify the loopback address in the *first* nameserver directive of the resolver configuration file so local clients can access the name server. Refer to “Step 6. Create the Loopback File” on page 716 for loopback address considerations.

The name server uses a private resolver that is different from the LE resolver used by other OS/390 UNIX socket programs. The name server has the following functional differences:

- The HFS file, `/etc/hosts`, is required for host table lookup if name services do not exist.
- Only the built-in translation table is used.

For a complete discussion of resolver configuration files, see *OS/390 SecureWay Communications Server: IP Migration* .

Creating the Syslog File

Syslog daemon (syslogd) is a server process that is typically started as one of the first processes in an OS/390 UNIX environment. Servers and stack components use syslogd for logging purposes and can also send trace information to syslogd. The named daemon logs messages to the syslog daemon. (For information about the syslog daemon, see *OS/390 SecureWay Communications Server: IP Diagnosis*.)

The name and location of your syslog file is specified in `/etc/syslog.conf`.

Querying Name Servers

This section describes how to use the onslookup command to query the name server.

Notes:

1. The onslookup command runs only from the OS/390 shell. To query the name server from TSO, use the nslookup command.
2. The host command is another way to query name servers from the OS/390 shell. See the *OS/390 SecureWay Communications Server: IP User’s Guide* for a list of host commands.

onslookup/nslookup Command

Note: For CS for OS/390 V2R8, well-known synonyms are provided for OS/390 UNIX shell commands. nslookup is provided as a synonym for the onslookup command.

The onslookup command lets you query the name server to perform the following tasks:

- Identifying the location of name servers.
- Examining the contents of a name server database
- Establishing the accessibility of name servers

onslookup has two modes of operation: interactive mode and command mode. In either mode, the address of the default name server comes from the resolver configuration data. In the sample data below, the default domain is raleigh.ibm.com, and the default name server is at 9.37.34.149. If that name server fails to respond, the one at 9.37.34.7 is used.

```
domain   raleigh.ibm.com
nameserver   9.37.34.149
nameserver   9.37.34.7
```

Entering the Interactive Mode

Interactive mode allows you to repetitively query one or more name servers for information about various hosts and domains, to display that information on your console, and, in some cases, to write response data to a file.

You can enter the interactive mode only under the following conditions:

- No arguments are supplied on command invocation. The default name server is used.
- The first argument is a hyphen, and the second argument is the host name or Internet address of a name server.

For a complete description of the onslookup interactive mode, see “onslookup/nslookup—Issuing Multiple Queries in Interactive Mode” on page 805.

Entering the Command Line Mode

The command line mode displays or stores the output from the query supplied as part of the invocation string and then exits.

To enter the command line mode, provide a complete query with the onslookup command invocation string.

For a complete description of the onslookup command line mode, see “onslookup/nslookup—Querying A Name Server in Command Mode” on page 803.

onslookup/nslookup Configuration

The configuration options of onslookup determine the operation and results of your name server queries. The values for onslookup options can be specified in more than one location, as shown in Table 27 on page 720. Table column headings are listed below. For example, column 1 lists onslookup options and column 2 lists options you can set in the `.onslookuprc` file.

1. onslookup command options
2. .onslookuprc file in the home directory
3. environment variable
4. /etc/resolv.conf
5. TCPIP.DATA configuration data set

Values specified as onslookup command options have priority over values specified in the .onslookuprc file, which have priority over the value specified by the environment variable, and so on. For example, the value specified by the all option in the onslookup command has priority over the value specified by the all option in the .onslookuprc file. Similarly, the value specified by ResolverTimeout in the /etc/resolv.conf file has priority over the value specified by ResolverTimeout in the TCPIP.DATA configuration data set.

The letters beside some settings indicate that the *terms* are functionally equivalent. For example, the term domain (the letter ″A″) is functionally equivalent to the terms DomainOrigin and LOCALDOMAIN. If two functionally equivalent settings are listed in the same file, the one listed last has priority. For example, if domain and DomainOrigin are both listed in the /etc/resolv.conf file and domain is listed first, the value specified by domain has priority.

Note: The /etc/resolv.conf file must contain all the data normally found in the TCPIP.DATA data set. If information is read from /etc/resolv.conf, then TCPIP.DATA is not used. To configure the /etc/resolv.conf file, you use certain directives. See “Resolv.conf Directives” on page 721 for detailed descriptions.

Table 27. Settings That Affect onslookup/nslookup Operation

onslookup Cmd Options	onslookupprc File in the Home Directory	Environment Variable	3	4	5
all	x	x			
class	x	x			
no[d2]	x	x			
[no]debug	x	x			
[no]defname	x	x			
domain (A)	x	x		x	
[no]ignoretc	x	x			
port (B)	x	x			
querytype	x	x			
[no]recurse	x	x			
retry (C)	x	x			
root	x	x			
[no]search	x	x			
srchlist (D)	x	x			
timeout (E)	x	x			
[no]vc (F)	x	x			
search (D)				x	
nameserver (G)				x	

Table 27. Settings That Affect onslookup/nslookup Operation (continued)

onslookup Cmd Options	onslookupprc File in the Home Directory	Environment Variable	3	4	5
sortlist				x	
options debug				x	
options ndots				x	
DomainOrigin (A)				x	x
NsInterAdd (G)				x	x
NsPortAddr (B)				x	x
ResolveVia (F)				x	x
ResolverTimeout (E)				x	x
ResolverUdpRetries (C)				x	x
LOCALDOMAIN (A)			x		

Resolv.conf Directives: The onslookup command uses a private resolver that is different than the LEresolver used by other OS/390 UNIX socket programs. The onslookup command requires the HFS file, `/etc/hosts` for host table lookup if name services do not exist. In addition, onslookup uses only the built-in translation table. For a complete discussion of resolver configuration files, see "Search Order and Configuration Files for TCP/IP Applications" on page 19.

Resolver configuration directives for `/etc/resolv.conf` are listed below.

domain *domain name*
Specifies the resolver's default domain if the host name is not fully qualified.

search *domain domain...*
Specifies an ordered list of domains for the resolver to search. The domain listed first is searched first.

nameserver *IP_address(es)*
Specifies the IP address or addresses of a particular name server to query. The addresses are queried in the order listed.

sortlist *subnet {/subnet_mask}...*
Specifies an ordered list of subnets and networks if the resolver receives more than one address as a result of a query. You can include one or more subnet masks or you can omit the mask to specify the entire network.

options *debug*
Starts the resolver debugging option.

options ndots: *minimum_number_of_dots*
Specifies the minimum number of dots an argument must have before the search list is applied. If the argument has less than the specified number of dots, the search list is appended to the name before any queries are sent. If the argument has the same number of dots or more, the query is sent first just as the user typed it. If a positive response is not received, subsequent queries are sent with the search list appended. The default is 1.

Configuring Host Resolvers: onslookup/nslookup Considerations

Programs that query a name server are called resolvers. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for

application programmers to perform queries. Under MVS, these routines are available in the TCP/IP application programming interface (API) for each supported language or LE for OS/390 UNIX Sockets API.

The onslookup command uses a private resolver that is different from the LE resolver used by other OS/390 UNIX socket programs. The onslookup command has the following functional differences:

- The HFS file, /etc/hosts, is required for host table lookup if name services do not exist.Following is a sample /etc/hosts file:

```
#
# OS/390 UNIX Resolver /etc/hosts file on mvss18oe.
#
# The format of this file is:
#
# Internet Address      Hostname   Aliases     # Comments
#
# Items are separated by any number of blanks and/or tabs.  A '#'
# indicates the beginning of a comment; characters up to the end of the
# line are not interpreted by routines which search this file.  Blank
# lines are allowed in this file.

9.24.104.126    mvs18oe mvsoe # OE host
192.168.210.1   mvs18an       # AnyNet MVS host
192.168.210.8   mypcaa        # AnyNet gw host
9.24.104.79     mypc          # A workstation
```

- Only the built-in translation table is used.

For a complete discussion of resolver configuration files, see “Search Order and Configuration Files for TCP/IP Applications” on page 19.

If the name server will run on the host being configured, you need to configure the *first* nameserver (or NsInterAddr) directive in the resolver configuration file as the loopback address (127.0.0.1 or any address in your home list).

Diagnosing Problems

This section describes four methods for diagnosing problems:

- Checking messages on the operators console.
- Checking the syslog messages.
- Using name server signals.
- Using the onslookup program.

These methods are discussed below. In addition to these methods, diagnosing problems for a dynamic zone can be done with nsupdate.

Checking Messages Sent to the Operators Console

Messages displayed on the operators console indicate the status of your DNS. Messages fall into four categories listed below.

- Name server initialization
- Name server initialization failure
- Name server initialization complete (always EZZ6475N)
- Name server termination

Check console messages regularly to identify problems.

Checking the Syslog Messages

Error messages are also displayed in the syslog output file, which is pointed to by `/etc/syslog.conf`. For descriptions of the syslog file and the syslog daemon, see *OS/390 SecureWay Communications Server: IP Diagnosis*.

Using Name Server Signals to Diagnose Problems

You can use name server signals to send messages to the named daemon. These signals control various functions that can be used to diagnose problems.

Diagnostic functions include the following:

- Enabling and disabling debug message logging
- Dumping the contents of the name server database
- Getting short status
- Logging queries

See "The named Daemon" on page 799 for descriptions of these tasks. For an explanation of possible output messages, refer to publications like *DNS and Bind* by Albitz and Liu.

Using onslookup/nslookup to Diagnose Problems

The onslookup program lets you query other name servers with the same query packet another name server would use. This is helpful in diagnosing lookup problems in TCP/IP.

To turn debugging on at level 1, enter the following commands from the OS/390 shell:

```
onslookup
set debug
```

The onslookup program shows timeouts and displays response packets. To turn the debug option off, enter the following command:

```
set nodebug
```

You can set the debugging option to level 2 by entering the following commands:

```
onslookup
set d2
```

The resolver shows the normal debugging information plus the query packets that were sent out. Turning on `d2` also turns on `debug`. Turning off `d2`, however, only turns off `d2` and `debug` remains on. To turn off both `d2` and `debug`, turn off `debug` by entering the subcommand and option `set nodebug`.

Connection Optimization in a Sysplex Domain

This section describes *connection optimization*, a technique that uses DNS for balancing IP connections and workload in a sysplex domain. It also dynamically manages the active and available list of IP addresses for resources in a sysplex domain.

Overview

Connection optimization uses DNS for distributing connections among hosts or server applications within a sysplex domain. A sysplex is a set of MVS systems communicating and cooperating with each other through multisystem hardware and software components.

In DNS terms, a sysplex is a subdomain that you add to your DNS name space. Name servers running within the sysplex perform name resolution. Resolvers query these name servers directly or indirectly through the name server authoritative for the resources in the sysplex domain.

Connection optimization extends the concept of a ″DNS host name″ to clusters, or groups of server applications or hosts. Server applications within the same group are considered to provide equivalent service. Connection optimization utilizes round-robin logic and load-based ordering to determine which addresses to return for a given cluster.

Connection optimization increases overall efficiency by favoring connections to systems with the most available resources and by avoiding unavailable sysplex resources. Addresses of the most available server applications or hosts are returned more frequently than the addresses of loaded server applications or hosts.

A connection-optimized sysplex domain is also scalable—that is, you can add servers and interface addresses dynamically to provide more service capacity. Client applications have dynamic access to the addresses of those servers, with no DNS restart or administration required.

Registration

To ensure maximum availability, server applications register with Workload Manager (WLM), which quantifies the availability of server resources within a sysplex. WLM must be configured in goal mode on all hosts within the sysplex. (See “Step 7: Configure WLM in Goal Mode” on page 736 for a description of this procedure.)

TCP/IP stacks also register with WLM. Additionally, they provide the active IP addresses. For a description of how IP addresses are associated with a sysplex domain name, see “Associating IP Addresses with the Sysplex Domain Name” on page 727. For a discussion of TCP/IP configuration issues, see “Configuring TCP/IP” on page 733.

When registering, server applications provide the following information:

- *Group name*. This is the name of a cluster of equivalent server applications in a sysplex. It is also the name within the sysplex domain that client applications use to access the server applications. To connect to *any* server application in a group, a client application uses the combination *group_name.sysplex_domain_name*.

  Note: The group name ″TCPIP″ and the group name for the sysplex domain (for example, ″mvsplex″) are reserved and cannot be used by server applications.
- *Server name*. This is the name of the server application instance. The server name must be unique among all servers that share the same group name. A server application instance can belong to more than one group.

- *Host name*. This is the host name of the TCP/IP stack on which the server application runs.

The sysplex domain name should be registered with the domain name server under a special address, 127.0.0.128. For details, see “Configuring the Name Server” on page 714.

Name Resolution

In connection optimization, a name server performs resolution for a name representing a cluster of hosts or server applications. Figure 32 depicts a sysplex domain called `mvsplex.mycorp.com`. The sysplex domain contains only the resources participating in the sysplex. Client applications append the domain name, `mycorp.com`, to all requests for name resolution.

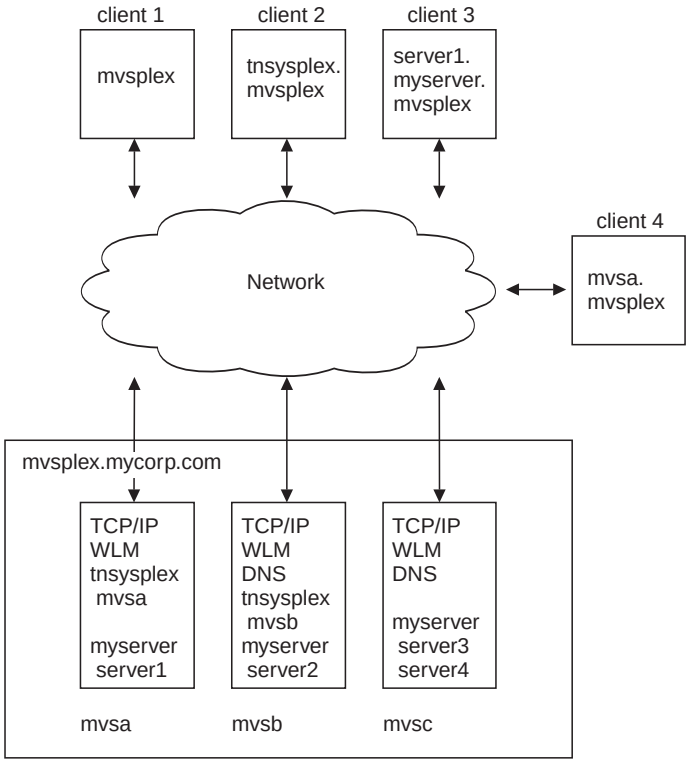


Figure 32. Name Resolution to a Sysplex

Each host system runs TCP/IP. Hosts `mvsb` and `mvsc` are also running the OS/390 UNIX sysplex name servers (`DNS`). `mvsb` is running the primary name server for the sysplex subdomain, and `mvsc` is running the secondary.

Each host is running one or more `myserver` server applications, and `mvsa` and `mvsb` are also running `tnsysplex` server applications. The group names (`tnsysplex` and `myserver`), the `tnsysplex` server names (`mvsa` and `mvsb`), and the `myserver` server names (for example, `server1`) are known to the name server through WLM registration by the respective application.

Note: `tnsysplex` represents a cluster of tn3270 server applications. Every instance of tn3270 running on a particular host registers using the same server name.

Four types of requests are shown in Figure 32 on page 725. Client Application 1 requests the services of any host on the system by providing only the name of the sysplex subdomain, `mvsplex`, to DNS for resolution. Client Application 2 requests the services of any `tnsysplex` server instance running in the sysplex. Client Application 3 requests a particular server instance in the `myserver` group, and Client Application 4 requests connection to a particular host.

In Figure 32 on page 725, only Client Applications 1 and 2 are candidates for connection optimization. Client Application 1 can connect to either `mvsa`, `mvsb`, or `mvsc`. Client Application 2 can connect to either `mvsa` or `mvsb` (but not `mvsc`). Client Applications 3 and 4, which can connect only to `mvsa`, are ineligible for connection optimization. They do, however, benefit from the round-robin selection process DNS uses to balance across available network interfaces.

Note: When using connection balancing via MVS clients (such as FTP), to avoid caching of IP addresses, during name resolution set the following environment variable:

```
export _EDC_IP_CACHE_ENTRIES=0
```

Generated Names vs. Statically Defined Names

All name servers use *statically defined* names. These are the names in the forward domain data file. As the DNS administrator for a name server using connection optimization, you must statically define the names of the hosts in the sysplex and the NS and SOA resource records in the forward domain data file. (For a description of sysplex forward domain data files, see “Sysplex Data Files” on page 735 .)

Note: The host names in the data files must match the host names specified in the stack’s TCPIP.DATA data set and should be 20 characters or less to ensure server uniqueness.

A name server using connection optimization also uses *generated names*. These are added dynamically to the domain name space as TCP/IP stacks and server applications in the sysplex register with WLM. The name server uses three types of generated names. In Figure 32 on page 725, the following generated names are used:

- The group name that is registered by the server applications (`tnsysplex` and `myserver`)
- The server name concatenated with the group name of each server application that registers with WLM in the sysplex (for example, `server1.myserver`)
- An alias for the sysplex domain name, `mvsplex`

The generated names become resources (or ″host″ names) within the sysplex domain, creating fully qualified domain names:

- Fully qualified group names (these are the connection balanced names):
  - `tnsysplex.mvsplex.mycorp.com`
  - `myserver.mvsplex.mycorp.com`
- Fully qualified server names:
  - `mvsa.tnsysplex.mvsplex.mycorp.com`
  - `server1.myserver.mvsplex.mycorp.com`
- Fully qualified alias name for the sysplex name `mvsplex.mycorp.com`:
  - `mvsplex.mvsplex.mycorp.com`

Associating IP Addresses with the Sysplex Domain Name: The sysplex domain name `mvsplex.mycorp.com` is associated with the *intersection* of the set of statically defined addresses associated with the stacks that are registered with WLM and the set of addresses associated with the adapters that are active on those stacks. The TCP/IP stack must be registered with WLM for this to occur. Figure 33 depicts this intersection.

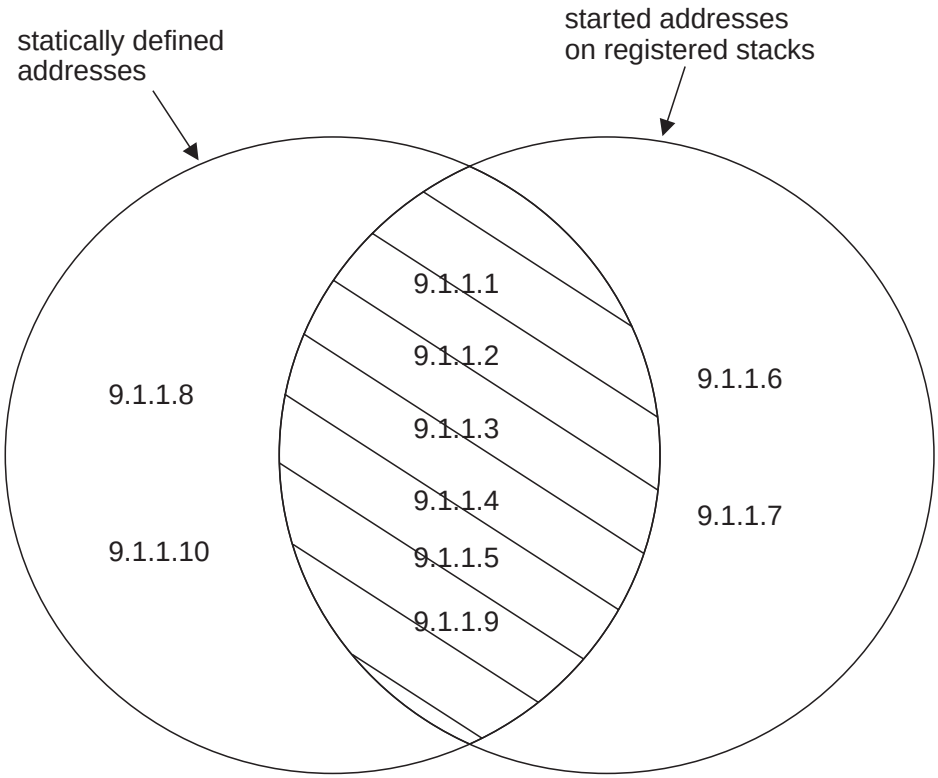


Figure 33. Address Association with mvsplex.mycorp.com

The hosts, `mvsa`, `mvsb`, and `mvsc` in the figure have the following addresses in the HOME statement in the *hlq*.PROFILE.TCPIP data set. Only certain adapters are active.

```
mvsa                    mvsb                    mvsc
9.1.1.1 (adapter active) 9.1.1.4 (adapter active)   9.1.1.8
9.1.1.2 (adapter active) 9.1.1.5 (adapter active)   9.1.1.9 (adapter active)
9.1.1.3 (adapter active) 9.1.1.6 (adapter active)   9.1.1.10
                         9.1.1.7 (adapter active)
```

The forward domain data file for the sysplex contains the following statically defined addresses:

```
mvsa            IN        A         9.1.1.1
                IN        A         9.1.1.2
                IN        A         9.1.1.3
mvsb            IN        A         9.1.1.4
                IN        A         9.1.1.5
mvsc            IN        A         9.1.1.8
                IN        A         9.1.1.9
                IN        A         9.1.1.10
```

Using the intersection of two sets lets the domain administrator selectively exclude certain IP addresses from use, such as 9.1.1.6 and 9.1.1.7 on mvsb, while distributing only active addresses to client applications. For instance, 9.1.1.8 would never be used since it is not active.

The process of associating IP addresses with server applications is similar to that for hosts. When a server application registers with WLM, the dynamically generated group name (for example, `myserver`) is added to the sysplex domain. If the stack on which the server application is running is registered with WLM, then the addresses associated with the group name are the intersection of the statically defined addresses for that stack and those addresses that are associated with adapters that are active. When the server application is replicated on other hosts in the sysplex, the addresses associated with the group name are the union of all intersection sets.

Detailed Example: The following example describes in detail how IP addresses become associated with a server application `myserver` running in the sysplex `mvsplex.mycorp.com`. As you read the following section, refer to Figure 34.

The example is described in terms of a *process*, beginning initially with no registered servers applications or stacks. The statically defined addresses associated with the `mvs` hosts and coded in the forward domain data file are the statically defined addresses listed earlier. As server applications and stacks register, the IP addresses associated with `myserver.mvsplex.mycorp.com` change. Figure 34 shows the conclusion of this process.

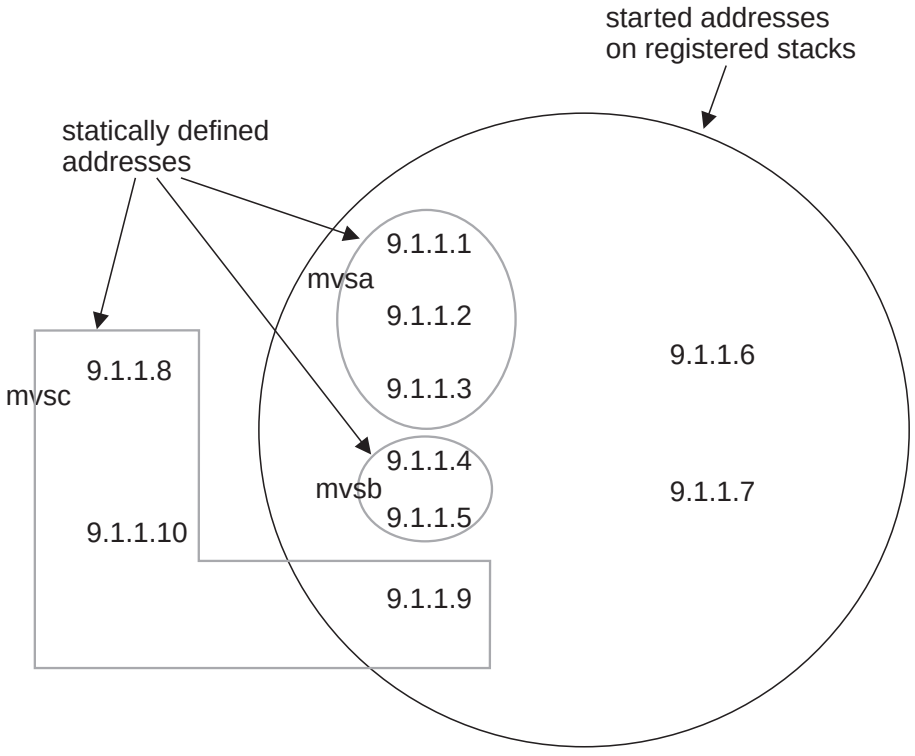


Figure 34. Address Association with myserver

Assume initially that a server application which registers with the group name myserver is running on mvsc, but that the stack on mvsc is not currently registered with WLM. The addresses associated with the name myserver.mvsplex.mycorp.com are 9.1.1.8, 9.1.1.9, and 9.1.1.10.

Next, assume that the stack on mvsc registers with WLM. The addresses associated with the name myserver.mvsplex.mycorp.com are reduced to the intersection of the statically defined addresses for the stack and those addresses that are associated with adapters that are active. The only address associated with group name myserver.mvsplex.mycorp.com is thus 9.1.1.9.

Now another instance of an equivalent server application registers with WLM on mvsb, but the stack on mvsb is *not* registered with WLM. All the statically defined addresses associated with mvsb are added to the set of addresses currently associated with the group name myserver.mvsplex.mycorp.com (9.1.1.4, 9.1.1.5, and 9.1.1.9). If the addresses on mvsb are active, the set of addresses associated with the group name does not change.

Similarly, if another instance of an equivalent server application registers with WLM on mvsa and if the addresses on mvsa are active, the set of addresses associated with the server application myserver.mvsplex.mycorp.com are those shown in Figure 34 on page 728 (and Figure 33 on page 727).

Note: The adapters associated with addresses 9.1.1.6 and 9.1.1.7 are not associated with the dynamically generated group name because they are not statically defined in the forward domain data file.

Connecting to a Particular Server Instance: The server name with which the server application registers is unique among all other server instances that share the same group name. Client applications may use the server name to bypass connection optimization. For example, client applications can bypass connection optimization if they are in the middle of a transaction with a server application, and the client/server session fails before the transaction completes. If the client application software has the ability to recognize this situation, the client can reconnect to the same server instance and complete the transaction.

The group name is prefaced with the server name, separated by a dot (server_name.group_name). If the group name is myserver and if the application instance on mvsc registered with WLM as myserver3, then client applications can connect to that particular instance using the name myserver3.myserver.mvsplex.mycorp.com. The address associated with this name (9.1.1.9) is a subset of the addresses currently associated with the group name myserver.mvsplex.mycorp.com that exist on stack mvsc.

Usage Considerations in a Connection Optimized Sysplex

Connection optimization extends the concept of a ″DNS host name″ to include a name that generically represents (1) all hosts in the sysplex (provided their stacks register with WLM) and (2) names that represent groups of equivalent server applications spread across the sysplex (provided those server applications register with WLM). The maximum benefit from connection optimization is realized when all stacks in the sysplex register with WLM and all TCP/IP server applications register with WLM, if they are capable of doing so.

You can, however, take advantage of connection optimization even when registration is not available. Consider the following scenarios.

TCP/IP Server Application Does Not Register: Suppose you want to use connection optimization for a particular TCP/IP server application, but it does not register with WLM. If the *stacks* that these applications are running on register with WLM, users can still use connection optimization with server applications by entering the sysplex domain name. A typical invocation might be the following:

```
tn3270 mvsplex
```

where tn3270 is the name of the invoked application and mvsplex is the sysplex domain name.

In this scenario, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Otherwise, connection optimization might result in connecting the client application to a host that is not running the server application. (`mvsc` in Figure 32 on page 725, for example, is not running the TN3270 server.) Depending on the client software, connection time-outs and connection retries might result.

One or More (or All) Stacks Do Not Register with WLM: Suppose you want to use connection optimization for a particular TCP/IP server application, but one or more (or all) stacks do not register with WLM because they do not support WLM registration or they are not configured to do so. Suppose also that the server application does not register with WLM. For the stacks that have not registered, only the statically defined addresses will be used.

In this scenario, a certain degree of connection optimization can occur between hosts whose stacks register with WLM if users enter the sysplex domain name (for example, `mvsplex` in Figure 32 on page 725.) In addition, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Connections will not be made to hosts whose stacks do not register.

Similarly, suppose that your application programmer develops a server application called `ourApp` that registers with the group name of `myserver`. Suppose also that one or more (or all) of the stacks on which the server application runs are not registered with WLM. A typical invocation might be the following:

```
ourApp myserver
```

Since one or more (or all) of the stacks are not registered with WLM, it is possible that an unusable IP address could be returned to the client because a stack has not reported the IP addresses that are active. In this case, only statically defined addresses are used for stacks that are not registered. If an unusable IP address is returned to the client, the connection times out, and depending on the client software, the client application may or may not retry the same or different address returned by the DNS query.

Considerations for Connection Balancing with Multiple Instances of TCP/IP on a Single Host System: Results of connection balancing in a multiple TCP/IP environment are not predictable in this release. Servers such as TN3270 that bind to a single instance of TCP/IP should work as expected as long as they provide the correct hostname in the WLM registration. Servers such as FTP which do not bind to a single instance of TCP/IP will have less predictable results. The hostname provided on the registration will link each server with a set of IP addresses for a single stack. Future releases of this product will allow the user to register with the sysplex name to avoid this problem.

Multiple Servers on the Same Port: If you are running multiple servers on the same port on the same TCP/IP instance (using SHAREPORT), only one of the servers should register with WLM.

Caching Issues

Proper distribution of server application addresses within a cluster requires DNS queries to be answered by the name server within the sysplex. For this reason, name servers that are located outside the sysplex cannot be configured as primary or secondary servers for the sysplex domain.

Name servers or resolvers outside the sysplex can prevent client application queries from reaching the sysplex name servers on subsequent requests if they use cached information. This is undesirable for connection optimization since the name servers and resolvers would not have up-to-date information about capacity and availability of the resources in the sysplex.

To disable other name servers from caching information about sysplex domain resources, a time-to-live (ttl) value of zero is returned by default. Note that some resolver and name server implementations do not support a ttl value of zero or anything less than an internally defined minimum (for example, 300 seconds).

Depending on the DNS and network configuration, the number of DNS queries on the network for the sysplex resources may increase. At the expense of reduced availability and load distribution information, administrators may choose a different default ttl for the sysplex resources by using the `-l` option when starting the sysplex name server.

Configuring a Sysplex Domain for Connection Optimization

Follow the steps below to configure name servers in a sysplex domain:

1. Identify server applications.
2. Configure server applications for WLM registration.
3. Choose sysplex name and identify name servers in the sysplex.
4. Update parent domain name server.
5. Configure the sysplex name servers.
6. Configure client applications.
7. Configure WLM in goal mode.

Each of these steps is explained below.

Step 1: Identify Server Applications

Identify the server applications you want to run in the sysplex. Refer to the product documentation for each application to determine if it supports registration with WLM for connection optimization. You can also modify your own application to register with WLM. (See “Registering Your Own Applications” on page 736.)

Candidate applications must have the following attributes:

- Client applications must use DNS for name resolution.
- Server applications must run in a single sysplex.
- Server applications within a specified group provide equivalent functions to their clients. That is, the client application receives the same services from any of the registered server applications.

- To be considered equivalent, all servers registering in a group must be listening on the same port.
- Client applications must use portmapper or a well-known port number to access the server application.

Note: Data Facility Storage Management Subsystem (DFSMS) restrictions do not currently allow sharing of HFS files in write mode.

Maximum benefits are attained when server applications have the following attributes:

- *Registration with WLM.* This feature allows WLM to track the availability of the registering server application and to allow clusters of servers on specified systems within the sysplex. Client application use of the sysplex domain name assumes that the server application is available on all hosts within the sysplex and the stacks running on the sysplex hosts are configured to register with WLM. Even if a server application does not register with WLM, it may still be able to take advantage of connection optimization under certain circumstances. (See "Usage Considerations in a Connection Optimized Sysplex" on page 729.)
- *Workloads.* The server application has system or network workloads sufficient enough to warrant load distribution. Determination of what is ″sufficient″ is subjective, but the value of balancing the incoming connections should outweigh the cost of the extra DNS queries on the network. See "Caching Issues" on page 731 for more information.
- *Session Length.* In most cases, server applications selected to use the connection optimization model should have long sessions. Since reduced caching in the network name servers causes additional network traffic, server applications with many short sessions may not be appropriate candidates.

Step 2: Configure Server Applications for WLM Registration

After you identify which server applications you want to run in the sysplex, you configure them for WLM registration. Typical configuration involves specification of the group name by which the application will be known. (You cannot use ″TCPIP″ or the sysplex domain name as a group name.) See "Registering Your Own Applications" on page 736 for information about how to register the applications you write. The following sections describe how to configure the tn3270 server and TCP/IP.

Configuring tn3270: To configure tn3270 servers for registration with WLM, use the WLMCLUSTERNAME and ENDWLMCLUSTERNAME statements to enter a unique sysplex server group name (or names) in the TELNETPARMS section of the *hlq*.PROFILE.TCPIP data set. (For a complete description of these statements, see "WLMCLUSTERNAME Statement" on page 428.)

Unique tn3270 server group names may be used to separate or direct client application requests to only those host systems supporting the target application. For example, a tn3270 server group name of `CICS3270` could be used on host systems actually running the target CICS application that the tn3270 clients access.

If necessary, you can modify the server group names using the VARY OBEYFILE command. (The VARY OBEYFILE command lets you make changes to the system operation and network configuration without stopping and restarting the TCP/IP address space.) If you use VARY OBEYFILE, however, you must specify *all* of the tn3270 parameters between the TELNETPARMS and ENDTELNETPARMS

statements (not just additions and deletions). For more information on OBEYFILE processing, see “VARY Command—TCPIP Address Space” on page 264.

Configuring TCP/IP: To register TCP/IP with WLM at startup, add the IPCONFIG statement with the SYSPLEXROUTING parameter to the *hlq*.PROFILE.TCPIP configuration data set. You can also register TCP/IP using VARY OBEYFILE by adding the IPCONFIG SYSPLEXROUTING statement to the obeyfile. For a complete description of this statement see, “IPCONFIG Statement” on page 213.

TCP/IP also registers addresses that are active for each stack. These addresses are the active, configured addresses in the `HOME` list for which a `START` has been processed. TCP/IP supports only 15 addresses per TCPIP instance.

To deregister TCP/IP, add IPCONFIG NOSYSPLEXROUTING to the obeyfile. (Although you can add this statement to the *hlq*.PROFILE.TCPIP data set, you typically deregister in the obeyfile.) If you use VARY OBEYFILE processing and you add IPCONFIG NOSYSPLEXROUTING, you must add IPCONFIG SYSPLEXROUTING on a subsequent VARY OBEYFILE command to reregister. Note that deregistration occurs automatically when TCP/IP terminates.

Note: Use the IPCONFIG SYSPLEXROUTING and IPCONFIG NOSYSPLEXROUTING statements only when your host is running as part of a sysplex.

Configuring the FTP Server: See “WLMCLUSTERNAME Statement” on page 607.

Configuring CICS: See *OS/390 eNetwork Communications Server IP CICS Sockets Guide Version 2 Release 5*.

Step 3: Choose Sysplex Name and Identify Name Servers

Identify a unique name with which DNS client applications can access the sysplex. This name may be the same as the configured sysplex name. Names must be 18 characters or less to accommodate WLM restrictions. The sysplex name becomes part of the fully qualified domain name. For example, a sysplex, `mvsplex`, in the domain `mycorp.com` has a fully qualified name of `mvsplex.mycorp.com`. (This is the domain name you specify in the primary directive that contains the `cluster` keyword in the named boot file.)

After you select a name for the sysplex, identify the master name servers. A sysplex should have only one primary name server, and it should have a secondary name server to provide redundancy. The secondary name server must run in the same sysplex as the primary name server. Both the primary and secondary name servers configured for connection optimization must run on hosts within the sysplex.

Step 4: Update Parent Domain Name Server

The parent domain name server is the primary name server authoritative for the domain that contains the sysplex subdomain. The parent domain name server may be the same name server as the sysplex name server.

Update the parent domain name server data files by entering the names and IP addresses of the name servers for the sysplex. Use NS (Name Server) resource records to enter the information.

For example, if the name of the domain in which the sysplex is located is `mycorp.com` and the name of the sysplex is `mvsplex` and the master name servers are running on hosts `mvsb` and `mvsc` in the sysplex, you would add the following records to the forward domain data file of the primary domain name server for `mycorp.com`:

```
mvsplex NS mvsb.mvsplex.mycorp.com.
        NS mvsc.mvsplex.mycorp.com.
mvsb.mvsplex.mycorp.com. A  9.67.116.201
                         A  9.67.116.206
                         A  9.67.116.208
mvsc.mvsplex.mycorp.com. A  9.67.116.203
                         A  9.67.116.207
                         A  9.67.116.210
```

This tells the parent domain name server that mvsb and mvsc are master servers for mvsplex. It does not tell the name server that mvsc is secondary, only that it is an additional master name server in the sysplex. The address (A) records are glue records. They enable remote name servers to contact the name servers in the sysplex.

In addition, it may be useful to add CNAME records for resources within the sysplex subdomain to avoid a name change in the client application and to prevent confusion for the end user. (See “Step 6: Configure Client Applications” on page 736.) For example, if client applications use a default domain name of mycorp.com, then requests for `tnsysplex.mycorp.com` can be delegated to the sysplex subdomain with the following resource record:

```
tnsysplex   CNAME   tnsysplex.mvsplex.mycorp.com.
```

Step 5: Configure the Sysplex Name Servers

When the TCP/IP stack is registered with WLM, the list of IP addresses made available to client applications includes the addresses that are common to those provided by TCP/IP and the list of application servers in the forward domain data file of the sysplex name server. See “Generated Names vs. Statically Defined Names” on page 726.

Note: The addresses returned to the client application depend upon several factors including whether all or some stacks in the sysplex are registered with WLM, the availability of the adapters assigned to the IP addresses, the names the client application uses for connection, and whether the servers are registered with WLM. See “Usage Considerations in a Connection Optimized Sysplex” on page 729.

The steps for configuring sysplex name servers are similar to those for configuring the name servers in an ordinary subdomain:

“Step 1. Specify Stack Affinity (Multiple Stack Environment)” on page 714.

“Step 2. Specify Port Ownership” on page 714.

“Step 3. Update the Name Server Start Procedure” on page 714.

“Step 4. Create the Domain Data Files (Primary Name Server Only)” on page 715. (See examples below.)

“Step 5. Create the Hints (Root Server) File” on page 716.

“Step 6. Create the Loopback File” on page 716.

“Step 7. Create the Boot File” on page 716. (See example below.)

“Step 8. Configure the Start Process” on page 717.

Sysplex Data Files: The data files must contain the ″A″ resource records for the host names internal to the sysplex. The host names must match the host names specified in the stack's TCPIP.DATA data set. For a discussion of how IP addresses are associated with a sysplex domain name, see "Associating IP Addresses with the Sysplex Domain Name" on page 727.

Note: You are encouraged to consider using VIPA addresses for your MVS hosts. If you use VIPA addresses, they are the only addresses you need to code in the forward domain data file.

Following is an example of a forward domain data file for a sysplex name server.

```
mvsplex.mycorp.com.  SOA mvsb.mvsplex.mycorp.com.
                              administrator@us.mycorp.com.  (
   1997061300  ; serial
   10800       ; refresh after 3 hours
   1800        ; retry 1/2 hour after failed zone transfer
   3600000     ; expire; length of time secondary keeps data unless refreshed
   259200 )    ; default TTL for static resource records = 3 days
;
;  Define nameservers
          IN  NS  mvsb.mvsplex.mycorp.com.
          IN  NS  mvsc.mvsplex.mycorp.com.
;
;  Define localhost
;
localhost IN  A   127.0.0.0
;
;  Hostnames specified here must match hostnames
;  specified in the stack's TCPIP.DATA file.
mvsa      IN  A     9.1.1.1
          IN  A     9.1.1.2
          IN  A     9.1.1.3
              TXT   "Text record"
              HINFO "3090" "OS/390R4"
mvsb      IN  A     9.1.1.4
          IN  A     9.1.1.5
          IN  A     9.1.1.6
          IN  A     9.1.1.7
mvsc      IN  A     9.1.1.8
          IN  A     9.1.1.9
          IN  A     9.1.1.10
```

Sysplex Boot Files: The boot file is the main configuration file for a name server. It points the name server to the domain data files, the loopback file, and the hints (root server) file. The contents and format of boot files for sysplex name servers are identical to those for the boot files of ordinary master servers with the exception of an additional keyword, `cluster`. This keyword is used only once in a boot file, at the end of either the primary or secondary directive to identify the sysplex domain.

The following is a sample boot file for the primary name server in a sysplex:

```
directory /etc/dnsdata
primary    mvsplex.mycorp.com         named.wlm.for    cluster
primary    113.67.9.in-addr.arpa      named.rev
primary    0.0.127.in-addr.arpa        named.lbk
cache      .                          named.ca
```

The file `named.wlm.for` identifies the forward domain data file for the primary name server in the sysplex.

The following is an example of a boot file for a secondary name server in a sysplex:

```
directory   /u/usr35/plex/secondary/zone1
secondary   mvsplex.tcp.raleigh.ibm.com   9.67.116.200 mvsplex.bak cluster
secondary   116.67.9.in-addr.arpa         9.67.116.200 mvsplex.rev
primary     0.0.127.in-addr.arpa          db.127.0.0
cache       .                             named.ca
```

Step 6: Configure Client Applications

In order for client applications to access a sysplex, you may have to change their resolver configuration files to use different default domain names or inform end users to use modified resource names for server access. (See “Name Resolution” on page 725 for the various names a client can specify.)

Step 7: Configure WLM in Goal Mode

All hosts in a sysplex must operate in goal mode for proper load balancing (splitting). If hosts are not in goal mode, they are treated with equal weight and round-robin selection is applied.

You can configure WLM in goal mode on a host by issuing the following MVS command:

```
F WLM,MODE=GOAL
```

You can also IPL in goal mode by omitting the `IPS=` keyword from your IEASYSxx parmlib member and from your IEASYS00 parmlib member. (See *OS/390 MVS Workload Management Services*.)

Registering Your Own Applications

You can register a server application with WLM using a C interface or an assembler interface. (See “Step 1: Identify Server Applications” on page 731.) The C function is invoked as follows:

```
extern long IWMDNREG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      char *wlm_user_data,
                      long *diag_code);
```

A sample header file, `iwmwdnsh.h`, comes with the product. Refer to the program directory for its location. The following definitions apply:

- *group_name* is the name client applications use. It can be up to 18 characters.
- *host_name* is the TCP/IP name of the host on which the server is running. (See *OS/390 C/C++ Programming Guide*.)
- *server_name* is a unique name that defines a particular instance of the server. It can be up to 8 characters.
- *netid* and *wlm_user_data* should be null pointers.

Return values and diag_code values are documented in *OS/390 MVS Workload Management Services* .

To register with a macro, use the IWMSRSRG macro. For a description of this macro, see *OS/390 MVS Workload Management Services*. When using this macro, note the following:

- *Location* contains the *group_name*
- *Network_ID* can be blank.
- *LUName* contains the *server_name*.
- *Host* contains the TCP/IP host name.

You can deregister a server application from WLM using a C interface or an assembler interface. Deregister whenever you do not want the server to receive additional client application connections. The C function is invoked as follows:

```
extern long IWMDNDRG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      long *diag_code);
```

To deregister with a macro, use the IWMSRDRS macro. For a description of this macro, see *OS/390 MVS Workload Management Services*.

For C interfaces for WLM registration and deregistration calls, see the sample registration file, `wlmreg.c`, that comes with the product. Refer to the program directory for the location of the file.

Dynamic IP

This section describes the purpose of Dynamic IP and its benefits. Also included is an introduction to the Dynamic IP components and an overview of design concepts.

Overview

To add a new workstation to an IP network, several parameters and a variety of information is required to configure the TCP/IP software. Network components, such as a domain name server, are also required. A new TCP/IP host would normally require the following information:

- IP address
- IP subnet mask
- Default router address
- Local hostname
- Domain name
- Name server addresss

Additional parameters, such as other server addresses, time zones, or protocol-specific configurations, might also be necessary.

Keeping track of that information in a large TCP/IP network is not an easy task for network administrators, especially if users or machines or both change their location frequently. IP address lists and domain name server databases have to be updated manually to keep track of any changes in the network.

From a user's point of view, a system administrator would have to be called to provide the necessary information to install a TCP/IP system. If the user moves to another location, this information must not be taken; the user will have to be assigned at least a new IP address if not a new hostname as well. Thus, users could potentially cause disorder in a TCP/IP network.

Even if workstations will be automatically installed using software distribution techniques, the TCP/IP configuration parameters have to be pre-assigned per distribution client.

The Bootstrap Protocol (BootP), as described in RFCs 951 and 1497, was introduced to the TCP/IP community in 1985 to provide automatic assignment of some TCP/IP configuration parameters to a new TCP/IP host. A table has to be maintained at BootP servers to enter information specific to any client that has been planned for installation. Typically, clients are identified by their LAN adapter's hardware address, which has to be known to the system administrator in charge of a BootP server before he can prepare a new client entry in the database. Even though some manufacturers nowadays put the adapter hardware address on a label on the backplane of their LAN adapters, this is a tedious process if many hosts have to be installed in a short period of time.

Objectives and Customer Benefits of Dynamic IP

To lessen the problems of having to manually update any centrally maintained information files and of having a user manually configure a TCP/IP workstation, the Dynamic Host Configuration Protocol (DHCP) has been designed and is described in RFCs 1533, 1534, 1541, and 1542. A DHCP server need not be pre-configured with a workstation's LAN address to submit the necessary TCP/IP configuration to it.

With DHCP in place, the assignment of IP addresses is a lot easier, but one problem persists—how would a domain name server learn about dynamically assigned IP addresses and hostnames so it can update its database accordingly? This can be solved by the Dynamic Domain Name Services (DDNS).

IBM is actively participating in the designs and implementations of DHCP and DDNS, and it has coined the term *Dynamic IP*. To summarize, the objectives of Dynamic IP and its benefits to TCP/IP system administrators and users are as follows:

- Provides automatic IP network access and host configuration
- Simplifies IP network administration
- Leverages existing IP network products and infrastructure
- Employs only open standards
- Allows customers to administer site-specific host environments
- Enables customized, location-sensitive parameter setups

Note: For further information on the Dynamic Host Configuration Protocol (DHCP) server, see "Configuring the DHCP Server for OS/390" on page 742.

Dynamic IP Components

Table 28 gives a brief description of the four types of network components that comprise Dynamic IP.

Table 28. DHCP Server Configuration

System Components	Description
Dynamic IP Hosts	Dynamic IP hosts contain DHCP client software and may contain Dynamic DNS client software. Together, they discover and cooperate with their DHCP and Dynamic DNS server counterparts in the network to automatically configure the hosts for network participation.

Table 28. DHCP Server Configuration (continued)

System Components	Description
DHCP Servers	DHCP servers provide the addresses and configuration information to DHCP and BootP clients on the network. DHCP servers contain information about the network configuration and about host operational parameters, as specified by the network administrator. DHCP server can also be configured to be the proxy for the DDNS client and issue the commands to update the Dynamic DNS server.
DDNS Servers	Dynamic DNS servers are a superset of today's static DNS servers. The dynamic enhancements enable client hosts to dynamically register their name and address mappings in the DNS tables directly, rather than having an administrator manually perform the updates.
BootP Relay Agents (or BootP Helpers)	BootP relay agents may be used in IP router products to pass information between DHCP clients and servers. BootP relays eliminate the need for having a DHCP server on each subnet to service broadcast requests from DHCP clients.

Administering Dynamic Domains

The DDNS server performs Dynamic DNS database updates in the appropriate domain file as the updates occur. Therefore, **do not edit domain files for dynamic zones while the DDNS server is running.** Furthermore, dynamic domains cannot be dynamically reinitialized with new configuration information using the traditional **nssig -HUP** command.

Also, note that if you enter comments into a domain file for a dynamic zone, the comments will be deleted when the first update to the domain is made. Domain file comments are not maintained because they would degrade the performance of the file update process.

You may change the configuration information for a dynamic domain in two different ways:

1. Manually, by editing the domain files **only after shutting down the DDNS server**.
2. Dynamically, by using **nsupdate** while the DDNS server is running.

For information on how to manually enter configuration information into domain files, refer to "Step 4. Create the Domain Data Files (Primary Name Server Only)" on page 715.

When you first set up your dynamic domain, you used **nsupdate -g** to create the zone key RSA key pair; **nsupdate -g** creates an entry in the /etc/ddns.dat file. The public key, the second key in the /etc/ddns.dat file entry, needs to be copied into the appropriate domain file. The zone private key stored in /etc/ddns.dat is used by **nsupdate** when signing update requests for the administrator of the zone. The server then examines the signature to identify update requests from the zone administrator versus those from ordinary hosts. The zone key gives the possessor the power to use **nsupdate** to create, modify, and delete any host's record in a dynamic domain.

Once the zone key information is generated and the DDNS server started, the administrator can take the /etc/ddns.dat file with him and administer the zone remotely using **nsupdate**.

Migrating an Existing DNS Configuration to Dynamic IP

Before you migrate a nameserver from static DNS to dynamic DNS, you should decide if you want to:

- Leave existing resource records as they are and allow new ones to be created and updated dynamically. This will allow existing systems to keep their hostnames, but they will not be able to update their resource records dynamically unless a system administrator deletes them.
- Delete all existing resource records and start with a dynamic domain from the beginning.

Follow the steps below to migrate existing DNS server configuration files to Dynamic IP:

1. Modify your existing DNS configuration files to resemble the files as shown in the previous example. In the case of a boot file you have to add the *dynamic secure* or *dynamic presecure* keywords to the *primary*statements for the authoritative DNS server that you are upgrading.
2. Use the NSUPDATE -g command to create the encryption keys. Copy the public key to the zone files.
3. Start the DDNS server.
4. If you have a DHCP server configured for DDNS updates, a new entry is needed in the DDNS.DAT file for each zone the DHCP server will update.

RSA Encryption

Since the OS/390 UNIX DDNS server and client products implement not only dynamic DNS but also DNS security functions, below is a brief explanation of cryptography. This section is courtesy of RSA Data Security, Inc., Redwood City, California, and has been modified slightly here for CS for OS/390.

Secret Key Cryptography

This method uses a secret key to encrypt a message. The same secret key must be used again to decrypt the message. This means that the key must be sent along with the message which exposes it to whoever might be eavesdropping on the conversation. Secret keys are very fast in terms of processing, and it is not easy to break them even though they are exposed through the communication process.

Public Key Cryptography

This method uses a combination of a modulus and a pair of exponents, called the public key and the private key. Exponents and modulus must be used together to encrypt or decrypt a message, but only the modulus and the public exponent are communicated since they are important to everyone who wants to send or receive encrypted messages using this method. The private exponent will never be publicly exposed. This ensures that no one else can decrypt messages that have been intended for a specified recipient, nor can anyone else disguise as that recipient to intercept a message.

Encryption and Authentication

Encryption means that a message will be scrambled before it can be sent over a communications link. The plain message itself will never be sent to ensure privacy. Authentication is used to ensure that a message has indeed originated from the source specified in the message, and that the message has not been altered in

transit. It additionally serves the purpose of non-repudiation, which means that whoever has digitally signed a message cannot claim later that he or she has not done so. In this case, the plain message itself will be sent since there is no need for privacy. The message will also be used to generate a digital signature by using one of the aforementioned cryptographic methods, preferably public keys.

Hash Functions

A hash function is a computation that takes a variable-size input and returns a fixed-size string, which is called the hash value. If the hash function is one-way, that means hard to invert, it is also called a message-digest function, and the result is called a message digest. The idea is that a digest represents concisely the longer message or document from which it was computed; one can think of a message digest as a digital fingerprint of the larger document.

The RSA Encryption Standard

This standard public key encryption method, along with the MD5 hash function, is used with the IBM DDNS product in CS for OS/390. The principle of the RSA algorithm is as follows:

1. Take two large primes, p and q.
2. Find their product n = p * q; n is called the modulus.
3. Choose a number, e, less than n and relatively prime to (p-1) * (q-1).
4. Find its inverse, d, mod (p-1) * (q-1), which means that e * d = 1 mod (p-1) * (q-1).

e and d are called the public and private exponents, respectively. The public key is the pair (n,e); the private key is d. The factors p and q must be kept secret or destroyed.

An example of RSA privacy (encryption) follows. Suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating:

```
c = m^e mod n
```

where e and n are Bob's public key. To decrypt, Bob also exponentiates:

```
m = c^d mod n
```

and recovers the original message, m; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

An example of RSA authentication follows. Suppose Alice wants to send a signed document, m, to Bob. Alice creates a digital signature s by exponentiating:

```
s = m^d mod n
```

where d and n belong to Alice's key pair. She sends s and m to Bob. To verify the signature, Bob exponentiates and checks that the message, m, is recovered:

```
m = s^e mod n
```

where e and n belong to Alice's public key.

Thus encryption and authentication take place without any sharing of private keys: each person uses only other people's public keys and his or her own private key. Anyone can send an encrypted message or verify a signed message, using only public keys, but only someone in possession of the correct private key can decrypt or sign a message.

To make encryption methods secure, a fairly large modulus should be chosen since it becomes increasingly difficult to break a large number into factors to determine the original primes. RSA uses a minimum length of 512 bits for the modulus, which would convert to a number with approximately 155 digits.

Due to security concerns, public key systems that use a key length of more than 512 bits must not be exported from the US.

For encryption, in reality, RSA is combined with a secret-key crypto system, such as DES, to encrypt a message by means of an RSA digital envelope. Data Encryption Standard (DES) is one of the most widely used secret key algorithms and was originally developed by IBM.

Suppose Alice wishes to send an encrypted message to Bob. She first encrypts the message with DES, using a randomly chosen DES key. Then she looks up Bob's public key and uses it to encrypt the DES key. The DES-encrypted message and the RSA-encrypted DES key together form the RSA digital envelope and are sent to Bob. Upon receiving the digital envelope, Bob decrypts the DES key with his private key, then uses the DES key to decrypt the message itself.

For authentication, in reality, RSA is combined with a hash function, such as MD5.

Suppose Alice wishes to send a signed message to Bob. She uses a hash function on the message to create a message digest, which serves as a digital fingerprint of the message. She then encrypts the message digest with her RSA private key; this is the digital signature, which she sends to Bob along with the message itself. Bob, upon receiving the message and signature, decrypts the signature with Alice's public key to recover the message digest. He then hashes the message with the same hash function Alice used and compares the result to the message digest decrypted from the signature. If they are exactly equal, the signature has been successfully verified, and he can then be confident that the message did indeed come from Alice. If, however, they are not equal, then the message either originated elsewhere or was altered after it was signed, and he rejects the message.

For authentication, the roles of the public and private keys are converse to their roles in encryption, where the public key is used to encrypt and the private key to decrypt. In practice, the public exponent is usually much smaller than the private exponent; this means that the verification of a signature is faster than the signing. This is desirable because a message or document will only be signed by an individual once, but the signature may be verified many times.

Configuring the DHCP Server for OS/390

Dynamic Host Configuration Protocol (DHCP) allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. The DHCP server controls whether the addresses it provides to clients are allocated permanently or are leased for a specific period. When a client is allocated a leased address, it must periodically request that the server revalidate the address and renew the lease.

The process of address allocation, leasing, and lease renewal are all handled dynamically by the DHCP client and server programs and are transparent to the end user.

DHCP defines three IP address allocation policies:

Dynamic
A DHCP server assigns a temporary, leased IP address to a DHCP client

Static A DHCP server administrator assigns a static, predefined address reserved for a specific DHCP client

Permanent
A DHCP server administrator assigns a permanent IP address to a DHCP client. No process of lease renewal is required.

Note: If your network uses routers or gateways, you need to ensure that they can be enabled as DHCP relay agents. Enabling the routers or gateways for DHCP allows the DHCP packets to be sent across the network to other LAN segments.

If you do not have routers that you can configure to be used as DHCP relay agents, you could:

- Use a UNIX system or RS/6000 system that has the necessary code to be configured to receive limited DHCP broadcasts. Then, forward those broadcast requests to the appropriate host server.
- Use a host server that is located on the same LAN segment as the IBM Network Stations. This would eliminate any need for routers or intermediate UNIX systems to pass on the broadcast requests of the IBM Network Stations.

For dynamic address allocation, a DHCP client that does not have a permanent lease must periodically request the renewal of its lease on its current IP address in order to keep using it. The process of renewing leased IP addresses occurs dynamically as part of the DHCP and is transparent to the user.

How Does DHCP Work?

DHCP allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. DHCP servers control whether the addresses they provide to clients are allocated permanently or are ″leased″ for a specific time period. When a client receives a leased address, it must periodically request that the server revalidate the address and renew the lease.

The DHCP client and server programs handle the processes of address allocation, leasing, and lease renewal.

To further explain how DHCP works, let's look at some frequently asked questions:

- How is configuration information acquired?
- How are leases renewed?
- What happens when a client moves out of its subnet?
- How are changes implemented in the network?

Acquiring Configuration Information: DHCP allows DHCP clients to obtain an IP address and other configuration information through a request process to a DHCP server. DHCP clients use RFC-architected messages to accept and use the options served them by the DHCP server. For example:

1. The client broadcasts a message (containing its client ID) announcing its presence and requesting an IP address (DHCPDISCOVER message) and desired options such as subnet mask, domain name server, domain name, and static route.

2. Optionally, if routers on the network are configured to forward DHCP and BOOTP messages (using BOOTP Relay), the broadcast message is forwarded to DHCP servers on the attached networks.
3. Each DHCP server that receives the client's DHCPDISCOVER message can send a DHCPOFFER message to the client offering an IP address (a server that does not want to serve a client can simply ignore the DHCPDISCOVER).

   The server checks the configuration file to see if it should assign a static or dynamic address to this client.

   In the case of a dynamic address, the server selects an address from the address pool, choosing the least recently used address. An address pool is a range of IP addresses to be leased to clients. In the case of a static address, the server uses a Client statement from the DHCP server configuration file to assign options to the client. Upon making the offer, the IBM DHCP server reserves the offered address.
4. The client receives the offer messages and selects the server it wants to use.
5. The client broadcasts a message indicating which server it selected and requesting use of the IP address offered by that server (DHCPREQUEST message).
6. If a server receives a DHCPREQUEST message indicating that the client has accepted the server's offer, the server marks the address as leased. If the server receives a DHCPREQUEST message indicating that the client has accepted an offer from a different server, the server returns the address it offered to the client to the available pool. If no message is received within a specified time, the server returns the address it offered to the client to the available pool. The selected server sends an acknowledgment which contains additional configuration information to the client (DHCPACK message).
7. The client determines whether the configuration information is valid. Upon receipt of a DHCPACK message, the DHCP client sends an Address Resolution Protocol (ARP) request to the supplied IP address to see if it is already in use. If it receives a response to the ARP request, the client declines (DHCPDECLINE message) the offer and initiates the process again. Otherwise, the client accepts the configuration information.
8. Accepting a valid lease, the client enters a BINDING state with the DHCP server, and proceeds to use the IP address and options.

To DHCP clients that request options, the DHCP server typically provides options that include subnet mask, domain name server, domain name, static route, class-identifier (which indicates a particular vendor), user class, and the name and path of the load image.

However, a DHCP client can request its own, unique set of options. For example, Windows NT® 3.5.1 DHCP clients are required to request options. The default set of client-requested DHCP options provided by IBM includes subnet mask, domain name server, domain name, and static route. For option descriptions, see "Specifying DHCP Options" on page 760.

Renewing Leases: The DHCP client keeps track of how much time is remaining on the lease. At a specified time prior to the expiration of the lease, usually when half of the lease time has passed, the client sends a renewal request, containing its current IP address and configuration information, to the leasing server. If the server responds with a lease offer, the DHCP client's lease is renewed.

If the DHCP server explicitly refuses the request, the DHCP client may continue to use the IP address until the lease time expires and then initiate the address request

process, including broadcasting the address request. If the server is unreachable, the client may continue to use the assigned address until the lease expires.

Moving a Client Out of its Subnet: One benefit of DHCP is the freedom it provides a client host to move from one subnet to another without having to know ahead of time what IP configuration information it needs on the new subnet. As long as the subnets to which a host relocates have access to a DHCP server, a DHCP client will automatically configure itself correctly to access those subnets.

For a DHCP client to reconfigure itself to access a new subnet, the client host must be rebooted. When a host restarts on a new subnet, the DHCP client may try to renew its old lease with the DHCP server which originally allocated the address. The server refuses to renew the request since the address is not valid on the new subnet. Receiving no server response or instructions from the DHCP server, the client initiates the IP address request process to obtain a new IP address and access the network.

Implementing Changes in the Network: With DHCP, you can make changes at the server, re-initialize the server, and distribute the changes to all the appropriate clients. A DHCP client retains DHCP option values assigned by the DHCP server for the duration of the lease. If you implement configuration changes at the server while a client is already up and running, those changes are not processed by the DHCP client until the client attempts to renew its lease or until it is restarted.

Setting Up a DHCP Network

The following sections contain information to help you in setting up your DHCP system:

- To create a scoped DHCP network, see “Creating a Scoped Network”.
- To start the DHCP server, see “Starting the DHCP Server” on page 746.
- For tips on maintaining a DHCP server, see “Maintaining the DHCP Server” on page 746.

The IBM DHCP server provides configuration information to clients based on statements contained in the server’s configuration file and based on information provided by the client. The server’s configuration file defines the policy for allocating IP addresses and other configuration parameters. The file is a ″map″ that the server uses to determine what information should be provided to the requesting client.

Before you start the DHCP server, create or modify the DHCP server configuration file.

Once the DHCP server is running, you can also make dynamic changes to the configuration by modifying the configuration file and using the DHCP Server Maintenance program to re-initialize the DHCP server.

Creating a Scoped Network: You create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients, while other configuration values are served only to certain clients. Serving different configuration information to clients is often based on network location, equipment vendor, or user characteristics.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.
- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially-defined options may be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.

For more information on obtaining information for a DHCP client, see "Maintaining the DHCP Server".

Handling Errors in Configuration Files: Configuring the server incorrectly causes few, if any, warning messages. The DHCP server normally runs even when it encounters errors in the configuration file. The server may ignore the incorrect data and may optionally post a message to its log.

Starting the DHCP Server:

Note: DHCPSD is installed in the /usr/lpp/tcpip/sbin directory.

To start the DHCP server, use a start procedure, or the following form of the **dhcpsd** command:

dhcpsd [-q|-v] [-f configFile]

-q Starts the server in **quiet** mode, which means that no banner is displayed when the server starts.

-v Starts the server in **verbose** mode. Causes messages dealing with client communication to print to screen.

-f configFile
Is the name of the DHCP server configuration file. By default, the server searches for a file called dhcpsd.cfg in the directory specified by the ETC environment variable.

When starting the DHCP server with a procedure (proc), the example start proc is found in the DHCP member of the installation partitioned data set SEZAINST.

Maintaining the DHCP Server:

Note: DADMIN is installed in the /usr/lpp/tcpip/sbin directory.

To maintain a running DHCP server, IBM provides the **dadmin** command to:

- Re-initialize a DHCP server by causing the server to re-read its configuration file
- Delete a lease
- Control server tracing
- Display client information
- Display IP address information
- Display server statistics

Note: Verbose mode provides additional information for debugging purposes. Verbose mode is allowed on any of the following **dadmin** command instances. Verbose is shown as a parameter in those instances where additional, more detailed information is of particular value.

Displaying dadmin Command Syntax: To display information about the command syntax, enter:

dadmin -?

Re-initializing the Running Server: If you make changes to the configuration file, you will need to re-initialize the running server to implement the changes. To re-initialize the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-i** [**-v**]

-h Specifies the host

host
The IP address or host name of the DHCP server. If no server is specified, the local server is assumed.

-i Re-initializes the specified server.

-v Executes the command in verbose mode.

Displaying Client Information: To display information for a client ID, use the following form of the **dadmin** command:

dadmin -c*value* [**-v**]

-c Requests information for one or more clients that match this client ID.

value
The client ID is a MAC address. For example, enter 004ac77150fc. Information is returned for any matching hardware type.

-v Executes the command in verbose mode.

Displaying IP Address Information: To display information for one IP address, use the following form of the **dadmin** command:

dadmin -q*n.n.n.n* [**-v**]

-q Requests the IP address information.

n.n.n.n
The IP address of the client.

-v Executes the command in verbose mode.

Querying an Address Pool: To display information for a pool of IP addresses, use the following form of the **dadmin** command:

dadmin -p*n.n.n.n* [**-v**]

-p Requests the address pool information.

n.n.n.n
The IP address of the address pool.

-v Executes the command in verbose mode.

Controlling Server Tracing: To start and stop tracing on the DHCP server, use the following form of the **dadmin** command:

dadmin -t*value* [**-v**]

-t Specifies server tracing.

value
The value is ON to start tracing or OFF to stop tracing.

-v Executes the command in verbose mode.

Displaying Server Statistics: To display statistics information about the pool of addresses administered by the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-n***value* [**-v**]

-h Specifies the host

host
The IP address of the DHCP server. If no host is specified, the local server is assumed.

-n Requests statistics for the server specified as *host*.

value
The value is a decimal integer indicating the number of intervals from 0 to 100. For example, a value of three returns a summary record that includes totals information, the current interval record, and the 3 most recent history records. A value of 0 returns a summary record of activity since the last summary.

-v Executes the command in verbose mode.

Statistics include:

- Discover packets processed
- Discover packets with no response
- Offers made
- Leases granted
- Negative acknowledgements (NAKs)
- Informs processed, including informs plus acknowledgements (ACKs)
- Renewals
- Releases
- BOOTP clients processed
- proxyARec updates attempted
- Unsupported packets
- Monitor requests processed

Deleting Leases: If you find that an assigned lease is not being used and you want to make the IP address available for allocation, you can delete the lease. You can

only delete one lease at a time. You will be prompted to confirm deletion of the lease. To delete the lease, use the following form of the **dadmin** command:

dadmin [**-f**] [**-v**] [[**-h**]*host*] **-d** *ip_address*

-f Forces deletion of the lease without prompting.

-v Executes the command in verbose mode.

-h

host
Specifies the IP address of the DHCP server. If no server is specified, the local server is assumed.

-d Deletes the lease for the specified IP address.

ip_address
The IP address for the lease to be deleted

Configuring the DHCP Server for the IBM Network Station Client: You can configure the DHCP server to be used by an IBM Network Station™. The DHCP server sets up the subnet and specifies the next bootstrap server. The IBM Network Station client can request information. The DHCP server should be configured to provide options that include subnet mask, router, domain name, and boot file name.

Changing the DHCP Configuration File

The name of the DHCP server configuration file is dhcpsd.cfg. The default location for dhcpsd.cfg is the \etc directory. In the configuration file, you create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients and other configuration values that are served only to certain clients. The information supplied to the clients is determined by the statements you use and the position of the statements in the configuration file.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.
- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially-defined options may be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.

- A sample DHCP server configuration file is installed in the HFS as /usr/lpp/tcpip/samples/dhcpsd.cfg.

Editing Tips

When editing the DHCP server configuration file, keep in mind the following:

- Comments must begin with a pound sign (#).
- Class and vendor names that include spaces must be surrounded by double quotes (″).
- Statement parameters are dependent on their position. If you omit a required parameter and enter a subsequent required parameter in a statement, the server recognizes that a parameter is missing, writes an error message to a log file, and continues to read the configuration file.
- A continuation character \ indicates that the information is continued on the next line. When used within a comment, the character is treated as part of the comment and is ignored as a continuation character.
- Braces are used to specify statements that are defined within other statements.
- If a parameter is specified in more than one place, the lowest level statement (which is the most specific) is used:
  - Statements specified outside braces are considered global and are used for all addresses served by this server unless the statement is overridden at a lower-defined level.
  - Parameters specified within braces under a statement such as a Subnet statement, are considered local and apply only to clients within the subnet.
  - Definition of a parameter in a class takes precedence over definition of the parameter in a subnet.
- Class statements are not allowed inside Client statements.
- Client statements are not allowed inside Option, Vendor, or Class statements.
- Subnet statements are not allowed inside Class or Client statements.
- Vendor statements are always defined at a global level.
- Keywords are not case sensitive. The capitalization that is used in this documentation is not required in the configuration file. However, use the convention that keywords start with a lower case letter and subsequent ″word″ subparts start with a capital letter. For example, a keyword is proxyARec.

DHCP Server Statements

ServerType Statement: To specify whether the server will operate only as a standard DHCP server, will perform both normal DHCP operations and PXE proxy DHCP operations, or will act only as a redirection server (PXE proxy DHCP server), use the following statement:

```
ServerType [DHCP | PXEDHCP | PXEPROXY]
```

The default value is `DHCP`, meaning the server will operate only as a standard DHCP server and will not interpret any PXE client extensions. The server will, however, still pass DHCP options and parameters to PXE clients.

PXEDHCP specifies that the server will perform both normal DHCP operations and PXE proxy DHCP operations. PXEPROXY indicates that the server will act only as a redirection server (PXE proxy DHCP server).

ImageServer Statement: To specify the siaddr field of the reply packet for a server acting as a PXE DHCP or PXE proxy only server, use the following statement:

```
ImageServer [ip address | hostname]
```

This statement separates the PXE siaddr field (the address of the BINL server) from the DHCP siaddr field. If the server is acting as a PXE DHCP server, it will use the ImageServer value to fill in the siaddr field of the reply packet. If the ImageServer statement is not specified, the siaddr field is left null for the PXE packet. A null value indicates that the BINL server resides on the same machine as the PXE proxy server. ImageServer is a global statement.

Log Statements: To enable logging by the server, use the following statements:

- Number of DHCP log files.

  `numLogFiles` *number_of_log_files*

  number_of_log_files is the maximum number of log files maintained.
- Size of DHCP log file.

  `logFileSize` *size_of_log_file*

  size_of_log_file is the size of the log file in kilobytes.
- Name of DHCP log file.

  `logFileName` *name_of_log_file*

  name_of_log_file is the name of the most recent log file.
- Type of log item.

  `logItem` *type_of_log_item*

  type_of_log_item is the type of the item to be logged. You should specify at least one log item. *type_of_log_item* can be:

  ```
  SYSERR
  OBJERR
  PROTERR
  WARNING
  EVENT
  ACTION
  INFO
  ACNTING
  TRACE
  ```

supportBootP Statement: To specify whether the server responds to requests from BOOTP clients, use the following statement:

```
supportBootP [YES | NO]
```

The default value is `NO`.

If this server previously supported BOOTP clients and has been reconfigured not to support BOOTP clients, the address binding for any BOOTP clients that was established before the reconfiguration is maintained until the BOOTP client sends another request (when it is restarting). At that time, the server does not respond, and the binding is removed.

Use this statement outside of braces.

supportUnlistedClients Statement: To specify whether the server responds to requests from DHCP clients other than those whose client IDs are specifically listed in this configuration file, use the following statement:

```
supportUnlistedClients [YES | NO]
```

The default is YES. If you specify NO, the server responds only to requests from DHCP clients that are listed (by client ID) in the configuration file.

For example:

```
client 6 10005aa4b9ab ANY
client 6 10a03ca5a7fb ANY
```

If the `supportUnlistedClients` statement is not specified or if you specify YES, the server responds to requests from any DHCP client. Use this option to limit access to addresses that are issued by this DHCP server. Listing the client IDs for all acceptable clients may take a long time.

Use this statement outside of braces.

bootStrapServer Statement: To specify whether the DHCP server specifies a bootstrap server for BOOTP clients, use the following statement:

`bootStrapServer` *address_of_bootstrap*

address_of_bootstrap is the IP address of the bootstrap server for the client.

This statement can appear at the global level, or within a Subnet, Class, or Client statement.

Option 67, Boot File Name: For clients who need a boot or must load images to initialize, use the DHCP Option 67 (Boot File Name) for the name of the boot file. The client downloads the image from the BOOTP server. For additional information about Option 67, see “Specifying DHCP Options” on page 760.

Lease Statements:

- To specify the default lease duration for the leases that are issued by the server, use the following statement:

  `leaseTimeDefault` *amount_of_default_lease_time*

  amount_of_default_lease_time is a decimal integer, followed by a space and a unit of time. The unit of time can be years, months, weeks, days, hours, minutes, or seconds.

  Default interval:
  24 hours (1440 minutes)

  Default unit:
  minute

  Minimum:
  180 seconds

  Maximum:
  -1, which is infinity

You can use this statement at the global level. To override this statement for a set of clients, use Option 51 (IP Address Lease Time) for a specific client, a class of clients, a subnet, or at the global level.

- To specify the interval at which the lease condition of all addresses in the address pool is examined, use:

  `leaseExpireInterval` *`interval_of_lease_time`*

  interval_of_lease_time is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed. The value that is specified should be less than the value for `leaseTimeDefault` to ensure that expired leases are returned to the pool in a timely manner.

  Default interval:
  1 minute

  Default unit:
  minute

  Minimum:
  15 seconds

  Maximum:
  12 hours

- To specify the maximum amount of time the server holds an offered address in reserve while waiting for a response from the client, use:

  `reservedTime` *`amount_of_time_reserved`*

  amount_of_time_reserved is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed.

  Default interval:
  5 minutes

  Default unit:
  minute

  Minimum:
  30 seconds

  Maximum:
  -1, which is infinity

Subnet Statement: Use the Subnet statement to specify configuration parameters for an address pool that is administered by a server. An address pool is a range of IP addresses to be leased to clients. If you configure subnets, you can set the lease time and other options for clients that use the address pool.

- Define a subnet.

  To define a subnet, use the following statement:

  `subnet` *`subnet_address`* [*`subnet_mask`*] *`subnet_range`* [(`alias=`*`subnet_name`*]

  Note: The DHCP Server Configuration program uses the parameters to the right of a left parenthesis. The DHCP server parses statements to the right of a left parenthesis as comments.

  subnet_address
  The address of this subnet, specified in dotted-decimal notation (for example, 192.67.48.0).

subnet_mask
The mask for the subnet, specified in dotted decimal notation or in integer format. A subnet mask divides the subnet address into a subnet portion and a host portion. If no value is entered for the subnet mask, the default is the class mask appropriate for an A, B, or C class network.

– Class A network - 255.0.0.0
– Class B network - 255.255.0.0
– Class C network - 255.255.255.0

A subnet mask can be expressed either in dotted-decimal notation or as an integer between 8 and 31. For example, you can enter a subnet mask as a dotted decimal notation of 255.255.240.0 or an integer format of 20. In subnet 192.67.48.0, a mask of 255.255.240.0 implies an address range from 192.67.48.001 to 192.67.63.254. The value 20 is the total number of 1s in a mask that is expressed in binary as 11111111.11111111.11110000.00000000.

Although not required, in most configurations the DHCP server should send Option 1 (Subnet Mask) to DHCP clients. Client operation may be unpredictable if the client does not receive subnet masks from the DHCP server and assumes a subnet mask that is not appropriate for the subnet.

subnet_range
All addresses, specified in dotted-decimal notation, to be administered to this subnet. The ranges should not overlap, for example, 192.67.48.1-192.67.48.128.

Notes:

1. In the range of addresses, do not include the address of the subnet and the address that is used for broadcast messages. For example, if the subnet address is 192.67.96.0 and the subnet mask is 255.255.240.0, do not include 192.67.96.0 and 192.67.111.255 in the range of addresses.
2. To exclude an IP address in a range of address, use the Client statement (see step 757).

(alias=*subnet_name*
is a symbolic name for ease in identifying a subnet.

- Define a subnet group.

  To define a subnet group, use the following statement:

  `subnet` *`subnet_address`* [*`subnet_mask`*] *`subnet_range`* [`label:`*`value[/priority]`*]

  label:*value[/priority]*
  identifies subnets that are grouped together on the same wire. *value[/priority]* is a string of 1 to 64 alphanumeric characters that identifies the subnet, followed by the priority in which this subnet's address pool is used. Do not uses spaces when specifying the label. More than one subnet can have the same identifier. *priority* is a positive integer, where 1 is a higher priority than 2. If you do not specify a priority, the highest priority is assigned. If two subnets have identical priority, the subnets within a label are processed based on the physical position in the configuration file.

  For example, the following two subnets are on the same wire:

  ```
  inOrder
  subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
  subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
  ```

- Serve IP addresses from multiple subnets.

To serve IP addresses from multiple subnets, use the `inOrder` or `balance` statement. The `inOrder` or `balance` statement is defined at a global level.

inOrder *subnet_labelslist*
subnet_labelslist is a list of labels in which each label identifies a subnet group. Each listed group is processed in order within that group. The subnet address pool with the highest priority within that group is completely exhausted before the subnet address pool with the next highest priority is used.

balance: *subnet_labelslist*
subnet_labelslist is a list of labels in which each label identifies a subnet group. The server provides the first IP address from the subnet that is first in the priority list, and subsequent IP addresses from each lesser-priority subnet, repeating the cycle until addresses are exhausted equally from all subnets.

The following is an example using the `inOrder` statement. Requests for subnet group WIRE1 exhaust addresses in subnet 192.67.48.0 (WIRE1/1) first, followed by subnet 192.67.49.0 (WIRE1/2). WIRE1 and WIRE3 are not related. Requests for subnet group WIRE3 exhaust addresses in subnet 192.67.50.0 (WIRE3/1) first, followed by subnet 192.67.51.0 (WIRE3/2), and then 192.67.50.0 (WIRE3/3), which has the same subnet address as WIRE3/1 but specifies a higher address range:

```
inOrder: WIRE3 WIRE1
subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
subnet 192.67.51.0 255.255.240.0 192.67.51.1-192.67.51.50 label:WIRE3/2
subnet 192.67.50.0 255.255.240.0 192.67.50.1-192.67.50.50 label:WIRE3/1
subnet 192.67.50.0 255.255.240.0 192.67.50.51-192.67.50.100 label:WIRE3/3
```

The following `balance` statement uses all IP addresses equally in WIRE1/3 and WIRE1/4:

```
balance: WIRE1
subnet 192.67.49.0 255.255.240.0 192.67.49.101-192.67.49.200 label:WIRE1/3
subnet 192.67.48.0 255.255.240.0 192.67.48.201-192.67.48.300 label:WIRE1/4
```

A sequence of `inOrder` or `balance` statements is cumulative. For example, the statements:

```
inOrder: WIRE1
inOrder: WIRE3
```

have the cumulative effect of the single statement:

```
inOrder: WIRE1 WIRE3
```

Note: To disable multiple subnets, comment out either the `balance` or `inOrder` processing statement or the priority.

Class Statement: Use the Class statement to specify configuration parameters for a user-defined group of clients that are administered by a server. You can use the Class statement at the global or subnet level. When the Class statement is specified within a Subnet statement, the server only serves clients in the class that are located in the specified subnet and request the class.

For example, to create a class that is called ″accounting″ that allows member hosts to use the LPR server (Option 9) at 192.67.123.2, do the following:

- At the DHCP server, define a class that is called ″accounting″ and set the LPR server for that class to 192.67.123.2
- At the client, configure the client to identify itself as belonging to the class ″accounting″.

When the client requests configuration information, the server sees that it belongs to the accounting class and provides configuration information that instructs the client to use the LPR server at 192.67.123.2. DHCP clients use Option 77 to indicate their class to DHCP servers.

To define a class, use the following statement:

`class` *`class_name`* [*`class_range`*]

class_name
The user-defined label that identifies the class. The class name is an ASCII string of up to 255 characters (for example, accounting). If the class name contains spaces, it must be surrounded by double quotes.

class_range
is a range of addresses. To specify a range of addresses, enter addresses in dotted-decimal notation, beginning with the lower end of the range, followed by a hyphen, then the upper end of the range, with no spaces between. For example, enter 192.17.32.1-192.17.32.128.

At a global level, a class cannot have a range. A range is allowed only when a class is defined within a subnet. The range can be a subset of the subnet range.

A client that requests an IP address from a class that has used all the addresses from its range is offered an IP address from the subnet range, if available. The client is offered the options associated with the class that has used all the addresses from its range.

To assign configuration parameters such as a lease time for all clients in a class, follow the Class statement with Option statements that are surrounded by braces. For more information about options, see“Specifying DHCP Options” on page 760 .

Client Statement: Use the Client statement to specify a unique set of options for a client. You can assign either a static address and configuration parameters, or configuration parameters. You can also use the Client statment to exclude an IP address from a range of available IP addresses. You can use the Client statement at the global, subnet, or class level.

- Define a client.

  `client` *`hw_type clientID client_ipaddr`* (`alias=`*`client_name`*

  hw_type
  A number that represents the hardware type of the client computer, required to decode the MAC address. For more information about hardware types, see “Hardware Types” on page 771.

  clientID
  Rither the hexadecimal MAC address, a string such as a domain name, or a name that is assigned to the client such as the host name. If you specify a string, you are required to enclose it in double quotes and specify 0 for the hardware type.

client_ipaddr
The DHCP client's IP address in dotted-decimal notation. *cleint_ipaddr* must contain an address if unlisted clients are not supported.

(alias=*client_name*
is a symbolic name for ease in identifying the client. Enter **alias=***client_name* immediately after a left parenthesis. This symbolic name appears in the display of the server configuration. If no name is entered, the MAC address is used.

For example, for a client (10005aa4b9ab), to reserve the static address 192.22.3.149 and also specify a lease time (Option 51) or 12 hours (43200 seconds) and a subnet mask (Option 1), use the following statement:

```
client 6 10005aa4b9ab 192.22.3.149
{
   option 51 43200
   option 1 255.255.255.0
}
```

- Specify options and assign any IP address to a client.

  To specify options for any IP address from the subnet, use the following `client` statement:

  `client` *hw_type clientID* `ANY`

  ANY
  specifies that any IP address can be assigned to the specific client ID.
- Exclude a client ID.

  To have the DHCP server exclude requests from a particular client ID, use the following `client` statement:

  `client` *hw_type clientID* `NONE`

  NONE
  specifies no IP address and no options are served to the specified client ID.

  For example:

  ```
  client 6 10005aa4b9ab NONE
  ```
- Exclude an IP address.

  To exclude one or more IP addresses from the pool of addresses available for lease, use the following statements:

  ```
  client 0 0 192.67.3.123
  client 0 0 192.67.3.222
  ```

  In this case, the hardware type and the client ID are 0. IP addresses 192.67.3.123 and 192.67.3.222 are excluded. You must specify a separate statement for each address you want excluded.
- Exclude a range of IP addresses.

  To exclude a range of IP addresses from the pool of addresses available for lease, specify many `client` statements.

  Each range of excluded addresses should contain no more than 10 addresses. Each excluded address results in a separate `client` statement in the configuration file. To exclude larger numbers of addresses, define subnets that do not include the addresses you want excluded. For example, to exclude addresses 50-75 in subnet 192.67.3.0, specify:

```
inOrder: WIRE1
subnet 192.67.3.0 255.255.240.0 192.67.3.1-192.67.3.49 label:WIRE1/1
subnet 192.67.3.0 255.255.240.0 192.67.3.76-192.67.3.100 label:WIRE1/2
```

Vendor Statement: To provide vendor configuration information to the DHCP clients in your network:

- Define a vendor and assign the appropriate configuration values. Unlike the Class statement, you cannot control the scope of the Vendor statement by its placement in the file. Use the Vendor statement only at the global level. Vendor statements within Subnet, Class, or Client statements are ignored. Options can be redefined in the vendor class.
- The DHCP client identifies itself to the DHCP server as belonging to a vendor class by sending Option 60 (Class Identifier) with a specific vendor name.
- The DHCP server recognizes that the client has a specific vendor and returns encapsulated Option 43 (Vendor-specific Information), which contains vendor-specific DHCP options and option values.

To define a vendor, use the following statement:

`vendor` *`vendor_name`* [`hex` *`value`*]

vendor_name
The user-defined label that identifies the vendor. The vendor name is an ASCII string of up to 255 characters (for example, ″IBM″). If the vendor name contains spaces, it must be surrounded by quotes (″).

[hex *value***]**
value must be specified either as an ASCII string or as hexadecimal in the hexadecimal ASCII string construct. For example:

```
hex "01 02 03"
```

For more information, see descriptions of Option 60 (Class-Identifier) in “Specifying DHCP Options” on page 760.

The vendor statement can also be specified in the DHCP server configuration file as a vendor statement followed by a pair of braces containing the options particular to this vendor. Within these braces, the usual option value encoding and decoding rules do not apply.

Statements Specifying Server Information:

- Querying in-use address.

  Before the server allocates an IP address, it PINGs the address to make sure that it is not already in use by a host on the network. The server places an in-use address in a special pool and allocates a different address.

  To specify the amount of time a DHCP server holds an in-use address in a special pool before returning the address to the active pool available for assignment, use the following statement:

  `usedIPAddressExpireInterval` *`in_use_time_value`*

  in_use_time_value is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. The default is 1000 seconds. If you do not specify a unit of time, minutes are assumed.

  Default interval:
  1000 seconds

Default unit:
minute

Minimum:
30 seconds

Maximum:
-1, which is infinity

- Transform canonical addresses.

  For 802.3 clients, use the canonical keyword to instruct the DHCP server to transform MAC addresses to canonical (byte starts with least significant bit) form. In most cases, you do not want the DHCP server to transform canonical addresses. MAC addresses of 802.3 clients are normally in canonical format on an 802.3 network. When 802.3 MAC addresses are transmitted across a transparent bridge, the bridge reformats the bits that identify an 802.3 client MAC address to a noncanonical (byte starts with most significant bit) form. When the bridge returns the MAC address to an 802.3 network, the bridge again reformats MAC addresses.

  To cause the DHCP server to transform MAC addresses, use the following statement:

  ```
  canonical [YES | NO]
  ```

  NO prevents the DHCP server from transforming MAC addresses. YES causes the DHCP server to transform MAC addresses. NO is the default value.

- Specify statistics snapshots.

  To specify the number of intervals that expire before the DHCP server takes a snapshot of statistics, use the following statement:

  `statisticSnapshot` *`number_of_intervals`*

  The length of each interval is determined by the `leaseExpireInterval` keyword. For example, a value of 3 collects statistics after a span of three intervals, where each interval has a length specified by the `leaseExpireInterval` keyword. If no value is specified, the server takes a snapshot of statistics at the end of every lease expire interval.

Additional Options: To assign additional configuration parameters, use the Option statement. All clients inherit all globally-defined options. A client that is defined within a Subnet statement inherits global options and options that are defined for that address pool. To assign configuration parameters for all clients in a subnet, follow the Subnet statement with Option statements that are surrounded by braces.

A Sample DHCP Server Configuration Files

The following is a sample DHCP configuration file:

```
logFileName dhcpsd.log
logFileSize 100
numLogFiles 4
logItem SYSERR
logItem ACNTING
logItem OBJERR
logItem EVENT
logItem PROTERR
logItem WARNING
logItem INFO
logItem TRACE
logItem ACTION
supportBootP yes
supportUnlistedClients NO
bootStrapServer 192.168.1.4
```

```
option 211 "nfs"
option 212 "10.1.1.2
option 213 "/usr/lpp/nstation/standard/configs/
option 214 "nfs"
option 67 /usr/lpp/nstation/standard/kernel

leaseTimeDefault 12 HOURS

option 15  mycompany.com

# Addresses 8.67.112.24 through 8.67.112.25 do not inherit
# options defined for 8.67.112.26 through 8.67.112.30

subnet 192.168.1.00 255.255.255.0 192.168.1.1-192.168.1.100
 {
  option 1 255.255.255.0
  option 3 10.1.1.1

  client 0 0 192.168.1.4
  client 0 0 192.168.1.5
 }
```

Specifying DHCP Options

DHCP allows you to specify options, also known as BOOTP vendor extensions, to provide additional configuration information to the client. RFC 2132 defines the options that you can use. Each option is identified by a numeric code.

Architected Options 0 though 127 and Option 255 are reserved for definition by the RFC. The DHCP server and the DHCP client use options in this set. The administrator can modfiy some architected options. Other options are for exclusive use by the client and server. The administrator cannot or should not configure the following options at the DHCP server:

- 52 (Option Overload)
- 53 (DHCP Message Type)
- 54 (Server Identifier)
- 55 (Parameter Request List)
- 56 (Message)
- 57 (Maximum DHCP Message Size)
- 60 (Class Identifier)

Options 128 through 254 represent options that can be defined by administrators to pass information to the DHCP client to implement site-specific configuration parameters. Additionally, IBM provides a set of IBM-specific options, such as Option 192 (TXT RR).

The format of user-defined options is:

`option` *code value*

code can be any option code 1 through 254. *value* must always be a string. At the server, it can be an ASCII string or a hexadecimal string. At the client, however, it always appears as a hexadecimal string that is passed to the processing program.

The server passes the specified value to the client. You must, however, create a program or command file to process the value.

Configuration File Option Data Formats: Table 29 contains a list of the data formats for the DHCP options:

Table 29. Data Formats for DHCP Options

Option	Description
IP Address	A single IP address in dotted-decimal notation
IP Addresses	One or more IP addresses in dotted-decimal notation separated by white spaces
IP Address Pair	Two IP addresses in dotted-decimal notation separated by a single colon
IP Address Pairs	One or more IP address pairs, each pair separated from another by a white space
Boolean	0 or 1
Byte	A decimal number between -128 and 127 (inclusive)
Unsigned Byte	A decimal number between 0 and 255 (inclusive). You cannot specify a negative value for an unsigned byte.
Unsigned Bytes	One or more decimal numbers between 0 and 255 (inclusive) separated by white spaces. You cannot specify a negative number for an unsigned byte.
Short	A decimal number between -32768 and 32767 (inclusive)
Unsigned Short	A decimal number between 0 and 65535 (inclusive). You cannot specify a negative number for an unsigned short.
Unsigned Shorts	One or more decimal numbers between 0 and 65535 (inclusive) separated by spaces. You cannot specify a negative number for an unsigned short.
Long	A decimal number between -2147483648 and 2147483647 (inclusive)
Unsigned Long	A decimal number between 0 and 4294967295 (inclusive). You cannot specify a negative number for an unsigned long.
String	A string of characters. If embedded spaces are used, the string must be enclosed in double quotes.
N/A	Indicates no specification is needed because the client generates this information

Option Categories: The DHCP options are divided into the following categories:

- Base Options (see Table 30)
- IP Layer Parameters per Host Options (see Table 31 on page 763)
- IP Layer Parameters per Interface Options (see Table 32 on page 764)
- Link Layer Parameters per Interface Options (see Table 33 on page 764)
- TCP Parameter Options (see Table 34 on page 765)
- Application and Service Parameter Options (see Table 35 on page 765)
- DHCP Extensions Options (see Table 36 on page 767)
- Load Balancing Options (see Table 37 on page 770)
- IBM-Specific Options (see Table 38 on page 771)

Table 30. Base Options

Option	Description	Data Format
1, Subnet Mask	The client's subnet mask, specified in 32-bit dotted decimal notation.	IP address

Table 30. Base Options (continued)

Option	Description	Data Format
2, Time Offset	The offset (in seconds) of the client's subnet from Universal Time Coordinated (UTC). The offset is a signed 32-bit integer.	Long
3, Router	IP addresses (in order of preference) of the routers on the client's subnet.	IP addresses
4, Timer Server	IP addresses (in order of preference) of the time servers available to the client.	IP addresses
5, Name Server	IP addresses (in order of preference) of the IEN 116 name servers available to the client.	IP addresses
6, Domain Name Server	IP addresses (in order of preference) of STD 13 name servers available to the client.	IP addresses
7, Log Server	IP addresses (in order of preference) of the MIT-LCS UDP log servers available to the client.	IP addresses
8, Cookie Server	IP addresses (in order of preference) of the Cookie or quote-of-the-day servers available to the client.	IP addresses
9, LPR Server	This option can be specified at both the DHCP client and DHCP server. However, if specified only at the DHCP client, the configuration will be incomplete. IP addresses (in order of preference) of the line printer servers available to the client. Option 9 eliminates the need for the client to specify the LPR_SERVER environment variable.	IP addresses
10, Impress Server	IP addresses (in order of preference) of the Imagn Impress servers available to the client.	IP addresses
11, Resource Location Server	IP addresses (in order of preference) of the Resource Location (RLP) servers available to the client. RLP servers allow clients to locate resources that provide a specified service, such as the Domain Name Server.	IP addresses
12, Host Name	This option can be specified at both the DHCP client and DHCP server. If the DHCP client does not provide a host name, the DHCP server does nothing with Option 12. Host name of the client (which may include the local domain name). The minimum length for the host name option is 1 octet and the maximum is 32 characters. See RFC 1035 for character set restrictions.	String

Table 30. Base Options (continued)

Option	Description	Data Format
13, Boot File Size	The length (in 512-octet blocks) of the default boot configuration file for the client.	Unsigned short
14, Merit Dump File	The path name of the merit dump file in which the client's core image is stored if the client crashes. The path is formatted as a character string consisting of characters from the Network Virtual Terminal (NVT) ASCII character set. The minimum length is 1 octet.	String
15, Domain Name	This option specifies the domain name that that client should use when resolving host names using the Domain Name System. This option can be specified at both the DHCP client and DHCP server.	String
16, Swap Server	IP address of the client's swap server.	IP address
17, Root Path	Path that contains the client's root disk. The path is formatted as a character string consisting of characters from the NVT ASCII character set. The minimum length is 1 octet.	String
18, Extensions Path	The extensions path option allows you to specify a string that can be used to identify a file that is retrievable using Trivial File Transfer Protocol (TFTP). The minimum length is 1 octet.	String

Table 31. Options Affecting the Operation of the IP Layer on a Per Host Basis

Option	Description	Data Format
19, IP Forwarding	Enable (1) or disable (0) forwarding by the client of its IP layer packets.	Boolean
20, Non-Local Source Routing	Enable (1) or disable (0) forwarding by the client of its IP layer datagrams with nonlocal source routes. The length is 1 octet.	Boolean
21, Policy Filter	IP address-net mask pair used to filter datagrams with nonlocal source routines. Any datagram whose next hop address does not match one of the filter pairs is discarded by the client. The minimum length for the policy filter option is 8 octets.	IP address pair
22, Maximum Datagram Reassembly Size	Maximum size datagram the client will reassemble. The minimum value is 576.	Unsigned short
23, Default IP Time-To-Live	Default time-to-live (TTL) the client uses on outgoing datagrams. TTL is an octet with a value between 1 and 255.	Unsigned byte

Table 31. Options Affecting the Operation of the IP Layer on a Per Host Basis (continued)

Option	Description	Data Format
24, Path MTU Aging Timeout	Timeout in seconds used to age Path Maximum Transmission Unit (MTU) values discovered by the mechanism that is described in RFC 1191.	Unsigned long
25, Path MTU Plateau Table	Table of MTU sizes to use in Path MTU discover as defined in RFC 1191. The minimum MTU value is 68. The minimum length for the path MTU plateau table option is 2 octets. The length must be a multiple of 2.	Unsigned shorts

Table 32. IP Layer Parameters Per Interface Options

Option	Description	Data Format
26, Interface MTU	Maximum Transmission Unit (MTU) to use on this interface. The minimum MTU value is 68.	Unsigned short
27, All Subnets are Local	Client assumes (1) or does not assume (0) that all subnets use the same Maximum Transmission Unit (MTU). A value of 0 means the client assumes that some subnets have smaller MTUs.	Boolean
28, Broadcast Address	Broadcast address used on the client's subnet.	IP address
29, Perform Mask Discovery	Client performs (1) or does not perform (0) subnet mask discovery using Internet Control Message Protocol (ICMP).	Boolean
30, Mask Supplier	Client responds (1) or does not respond (0) to subnet mask requests using Internet Control Message Protocol (ICMP).	Boolean
31, Perform Router Discovery	Client solicits (1) or does not solicit (0) routers using router discovery as defined in RFC 1256.	Boolean
32, Router Solicitation Address	Address to which a client transmits router solicitation requests.	IP address
33, Static Route	Static routes (destination address-router pairs in order of preference) the client installs in its routing cache. The first address is the destination address and the second address is the router for the destination. Do not specify 0.0.0.0 as a default route destination.	IP address pairs

Table 33. Link Layer Parameters Per Interface Options

Option	Description	Data Format
34, Trailer Encapsulation	Client negotiates (1) or does not negotiate (0) the use of trailers (RFC 893) when using Address Resolution Protocol (ARP).	Boolean

Table 33. Link Layer Parameters Per Interface Options (continued)

Option	Description	Data Format
35, ARP Cache Timeout	Timeout in seconds for Address Resolution Protocol (ARP) cache entries.	Unsigned long
36, Ethernet Encapsulation	For an Ethernet interface, the client uses IEEE 802.3 (1) Ethernet encapsulation described in RFC 1042 or Ethernet V2 (0) encapsulation described in RFC 894.	Boolean

Table 34. TCP Parameter Options

Option	Description	Data Format
37, TCP Default TTL	Default time-to-live (TTL) the client uses for sending TCP segments.	Unsigned byte
38, TCP Keep-alive Interval	Interval in seconds the client waits before sending a keep-alive message on a TCP connection. A value of 0 indicates that the client does not send keep-alive messages unless requested by the application.	Unsigned long
39, TCP Keep-alive Garbage	Client sends (1) or does not send (0) TCP keep-alive messages that contain an octet of garbage for compatibility with previous implementations.	Boolean

Table 35. Application and Service Parameter Options

Option	Description	Data Format
40, Network Information Service Domain	The client's Network Information Service (NIS) domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set. The minimum length is 1 octet.	String
41, Network Information Servers	IP addresses (in order of preference) of Network Information Service (NIS) servers available to the client.	IP addresses
42, Network Time Protocol Servers	IP addresses (in order of preference) of Network Time Protocol (NTP) servers available to the client.	IP addresses
43, Vendor-Specific Information	Option 43 is specified only at the DHCP server, which returns this option as an encapsulated packet to a client that sends Option 60 (Class Identifier). Clients and servers use this information option to exchange vendor-specific information. This option allows for expansion of the number of options that can be supported.	String
44, NetBIOS over TCP/IP Name Server	IP addresses (in order of preference) of NetBIOS name servers (NBNS) available to the client.	IP addresses

Table 35. Application and Service Parameter Options (continued)

Option	Description	Data Format
45, NetBIOS over TCP/IP Datagram Distribution Server	IP addresses (in order of preference) of NetBIOS datagram distribution (NBDD) name servers available to the client.	IP addresses
46, NetBIOS over TCP/IP Node Type	Node type used for NetBIOS over TCP/IP configurable clients as described in RFC 1001/1002. Values to specify client types include: **Value** **Node Type** **0x1** B-node **0x2** P-node **0x4** M-node **0x8** H-node	Unsigned byte
47, NetBIOS over TCP/IP Scope	NetBIOS over TCP/IP scope parameter for the client, as specified in RFC 1001/1002. The minimum length is 1 octet.	Unsigned byte
48, X Window System Font Server	IP addresses (in order of preference) of X Window System font servers available to the client.	IP addresses
49, X Window System Display Manager	IP addresses (in order of preference) of systems running X Window System Display Manager available to the client.	IP addresses
64, NIS Domain Name	Network Information Service (NIS)+ client domain name. The domain is formatted as a character string consisting of characters from the NVT ASCII character set. Its minimum length is 1.	String
65, NIS Servers	IP addresses (in order of preference) of Network Information Service (NIS)+ servers available to the client.	IP addresses
68, Home Address	IP addresses (in order of preference) of the mobile IP home agents available to the client. This option enables a mobile host to derive a mobile home address and determine the subnet mask for the home network. The usual length is four octets, containing a single home agent's home address.	IP addresses
69, SMTP Servers	IP addresses (in order of preference) of the Simple Mail Transfer Protocol (SMTP) servers available to the client.	IP addresses
70, POP3 Server	IP addresses (in order of preference) of the Post Office Protocol (POP) servers available to the client.	IP addresses

Table 35. Application and Service Parameter Options (continued)

Option	Description	Data Format
71, NNTP Server	IP addresses (in order of preference) of the Network News Transfer Protocol (NNTP) servers available to the client, for example: `option 71 192.24.112.2`	IP addresses
72, WWW Server	IP addresses (in order of preference) of the World Wide Web (WWW) servers available to the client.	IP addresses
73, Finger Server	IP addresses (in order of preference) of the Finger servers available to the client.	IP addresses
74, IRC Server	IP addresses (in order of preference) of the Internet Relay Chat (IRC) servers available to the client.	IP addresses
75, StreetTalk Server	IP addresses (in order of preference) of the StreetTalk servers available to the client.	IP addresses
76, STDA Server	IP addresses (in order of preference) of the StreetTalk Directory Assistance servers available to the client.	IP addresses

Table 36. DHCP Extensions Options

Option	Description	Data Format
50, Requested IP Address	This option is specified only at the DHCP client. The DHCP server can refuse a DHCP client request for a specific IP address. Allows the client to request (DHCPDISCOVER) a particular IP address.	N/A
51, IP Address Lease Time	This option can be specified at both the DHCP client and DHCP server. The DHCP client can use Option 51 to override the defaultLeaseInterval value the DHCP server offers. Allows the client to request (DHCPDISCOVER or DHCPREQUEST) a lease time for an IP address. In a reply (DHCPOFFER), a DHCP server uses this option to offer a lease time.	Unsigned long
66, Server Name	Trivial File Transfer Protocol (TFTP) server name used when the ″sname″ field in the DHCP header has been used for DHCP options.	String

Table 36. DHCP Extensions Options (continued)

Option	Description	Data Format
67, Boot File Name	Name of the boot file when the 'file' field in the DHCP header has been used for DHCP options. The minimum length is 1. **Note:** Use this option to pass a boot file name to a DHCP client. The boot file name is required to contain the fully-qualified path name and be less than 128 characters in length, for example: `option 67 c:\usr\lpp\nstation\ standard\kernel` This file contains information that can be interpreted in the same way as the 64-octet vendor-extension field within the BOOTP response, with the exception that the file length is limited to 128 characters by the BOOTP header.	String
58, Renewal (T1) Time Value	Interval in seconds between the time the server assigns an address and the time the client moces to the renewing state.	Unsigned long
59, Rebinding (T2) Time Value	Interval in seconds between the time the server assigns an address and the time the client enters the rebinding state.	Unsigned long
60, Vendor Class-Identifier	The DHCP client optionally uses this option to identify the vendor type and configuration of the client. For example, the identifier may encode the client's vendor-specific hardware configuration. The information is a string of **n** octets interpreted by servers. Vendors may choose to define specific vendor class identifiers to convey particular configuration or other identification information about a client. Servers not equipped to interpret the class-specific information sent by a client must ignore it. The minimum length is 1 octet.	N/A

Table 36. DHCP Extensions Options (continued)

Option	Description	Data Format
61, Client-Identifier	DHCP clients use this option to specify their unique identifier. DHCP servers use this value to index their database of address bindings. This value must be unique for all clients in an administrative domain. The Client-Identifier may consist of type-value pairs. For example, it may consist of a hardware type and hardware address. In this case, the type field should be one of the ARP hardware types defined in STD2 [22]. A hardware type of 0 (zero) should be used when the value field contains an identifier other than a hardware address (for example. a fully qualified domain name). For correct identification of clients, each client's client-identifier must be unique among the client-identifiers used on the subnet to which the client is attached. Vendors and system administrators are responsible for choosing client-identifiers that meet this requirement for uniqueness.	String
62, NetWare/IP Domain Name	Netware/IP Domain Name. The minimum length is 1 octet and the maximum length is 255.	String
63, NetWare/IP	A general purpose option code used to convey all the NetWare/IP related information except for the NetWare/IP domain name. A number of NetWare/IP suboptions are conveyed using this option code. The minimum length is 1 and the maximum length is 255.	String
77, User Class	DHCP clients use Option 77 to indicate to DHCP servers what class the host is a member of.	String

Table 36. DHCP Extensions Options (continued)

Option	Description	Data Format
78, Directory Agent	The Dynamic Host Configuration Protocol provides a framework for passing configuration information to hosts on a TCP/IP network. Entities using the Service Location Protocol need to find out the address of Directory Agents in order to transact messages. In certain other instances, they may need to discover the correct scope and naming authority to be used in conjunction with the service attributes and URLS, which are exchanged using the Service Location Protocol. A directory agent has a particular scope and may have knowledge about schemes defined by a particular naming authority.	IP addresses
79, Service Scope	This extension indicates a scope that should be used by a service agent, when responding to Service Request messages as specified by the Service Location Protocol.	String
80, Naming Authority	This extension indicates a naming authority, which specifies the syntax for schemes that may be used in URLs for use by entities with the Service Location Protocol.	String

DHCP Load Balancing Options: To configure DHCP for load balancing on an OS/390 server, you must define Options 211 through 214. You define DHCP classes on the subnet level. Because you configure the load balancing values on the DHCP class, only Network Stations can use them. If you have any other devices using DHCP on that same subnet, they will not be affected.

Table 37 lists and describes the DHCP load balancing options.

Table 37. DHCP Load Balancing Options

Option	Description	Data Format
211 (Base Code Server Protocol)	Protocol to use for Option 66 (Base Code Server).	String
212 (Terminal Configuration Server)	Terminal configuration server IP address of name. You can specify up to two addresses separated by a blank.	String
213 (Terminal Configuration Path)	Configuration file path name for Option 212 (Terminal Configuration Server). You can specify up to two paths separated by a blank.	String
214 (Terminal Configuration Protocol)	Protocol to use for Option 212 (Terminal Configuration Server). You can specify up to two values separated by a blank.	String

An example of using load balancing options follows:

```
subnet __line
 {
  option 211 "nfs"
  option 212 "192.5.179.25"
  option 213 "/usr/lpp/nstation/standard/configs/"
  options 214 "nfs"
 }
 class IBMNSM 1.0.0
 class IBMNSM 2.0.0
 vendor IBM Network Station
```

IBM-Specific Options: IBM provides a set of IBM-specific options that fall within nonarchitected Options 128-254, which administrators use to implement site-specific configuration parameters.

Additionally, architected Option 43 allows the definition of encapsulated, vendor-specific options. IBM Corporation, for example, has added the following IBM-specific options, denoted by an IBM-specific file in Option 60.

Table 38. IBM-Specific Options

Option	Description	Data Format
200, LPR Printer	Eliminates the need for the client to specify the LPR_PRINTER environment variable, which can be the name of a device such as LPT1 or a printer name (queue name) such as Printer. For example, `option 200 "lpt1"` An OS/2 client stores the updated option value in the TCPOS2.INI file. The length is 1 octet.	String

Options that are encapsulated as vendor-specific information must be carefully defined and documented to permit interoperability between clients and servers from different vendors. Vendors that define vendor-specific information must do the following:

- Document those options in the form that is specified in RFC 2132.
- Choose to represent those options either in data types already defined for DHCP options or in other well-defined data types.
- Choose options that can be readily encoded in configuration files for exchange with servers that are provided by other vendors.
- Be readily supportable by all servers.

Servers that are not equipped to interpret the vendor-specific information that is sent by a client must ignore it.

Clients that do not receive desired vendor-specific information should make an attempt to operate without it. Refer to RFC 2131 and RFC 2132 for additional information about this option.

Hardware Types

The following is a list of hardware types you can specify on the Client statement:

Type	Description
0	Unspecified. If you specify a symbolic name for the client ID, specify 0 for the hardware type.
1	Ethernet (100MB)
6	IEEE 802 Networks (which include 802.5 token ring)

Configuring DHCP Server as DDNS Client Proxy

The DDNS proxy A-record update feature supports clients that do one of the following:

- Use DHCP option 81 to communicate their host name information to a DHCP server
- Send some or all of their host name information in DHCP option 12 and option 15
- Have their host name information defined to the DHCP server by an administrator

The proxyARec update feature allows non-Dynamic IP clients (clients which have DHCP client capability but not DDNS capability) to effectively participate in the Dynamic IP system. This alternative is well-suited for network environments where client hosts are not mobile or do not change administrative domains.

New keywords in the DHCP server configuration file instruct the DHCP server how to interact with the DDNS server using A records on behalf of DHCP clients. The new keywords are:

proxyARec
For more information on enabling DHCP server A record updates, see "Enabling Dynamic A Record Updates".

updateDNSA
For more information on enabling A record updates, see "Specifying the Keyword for A Record Updates" on page 773.

updateDNSP
For more information on enabling PTR record updates, see "Specifying the Keyword for PTR Record Updates" on page 775.

Enabling Dynamic A Record Updates

Once the DHCP server reads the configuration file and is instructed how and when to perform a DDNS proxy A record update, the following rules of precedence are used to derive the host name data to be used:

1. As a first priority, option 81 is included in the DHCP client request and indicates that an A record update should be performed at the DDNS server on behalf of the client.
2. Either option 12 or option 15, or both, are included in a DHCP client request.
3. Either option 12 or option 15, or both, are defined in the DHCP server configuration file.

The data used in the DDNS update packet is derived using one or more of the above means until a fully-qualified host name data is established. For example, in the case of Microsoft® Windows® 3.11+ and Windows 95 clients, only option 12, the Microsoft Windows client computer name, is included in DHCP lease requests. In

this case, the administrator can define an option 15, domain name, to use with the host name provided by a client's option 12 to obtain the host's fully-qualified name.

The proxy A-record update feature works with any client which is capable of including host name info in option 81 and/or options 12 and 15. Otherwise, the DHCP client host, as identified by its LAN adapter's MAC address, and its host name information can be specified entirely in the DHCP server's configuration file.

Note: If you use DDNS proxy A record support for subnets which include some hosts capable of updating their own DDNS A-records such as IBM OS/2® Warp Dynamic IP clients, you may wish to disable those client's own DDNS update feature by commenting out updateDNSA and UpdateDNSTxt keywords in the client's DHCPCD.CFG configuration file. If you do not disable the dynamic A record updates at the clients, the proxy A record updates attempted at the DHCP server for those clients will fail because the DHCP client, rather than the DHCP server, will own the client's entry on the DDNS database.

Specify the proxyARec keyword to enable the DHCP server to dynamically update an A record mapping (host name-to-IP-address) on a DDNS server for clients unwilling or unable to update their A records. The format of the proxyARec keyword is:

```
proxyARec
```

The DHCP server performs updates of the client's A record regardless of the client's client ID value.

The proxyARec keyword is required to be enabled globally or within the subnet, class, or client level. The updateDNSA keyword must also be specified. If the proxyARec keyword is not specified, clients will not have their A records updated.

Specify the proxyARec statement within braces to indicate the information applies only to the subnet, class, or clients that meet the criteria of the preceding conditional statement. To globally assign proxyARec, specify the keyword outside any braces.

Specifying the Keyword for A Record Updates

To specify how the server updates A records for a non-IBM IP client, use keyword:

```
updateDNSA command_string
```

The following is a typical updateDNSA string issued by the DHCP server:

```
updateDNSA "nsupdate -f -h%s -d%s -s"d;a;*;a;a;%s;s;%s;3110400;q""
```

This default string is normally adequate. Typical changes to the command string are to modify the value of an expiration time (such as 3110400 seconds) beyond the A record expiration. Assuming proxyARec is specified, this example instructs the DDNS name server to delete all A records for this host name and add a record that maps the host name to the specified IP address for the specified lease time.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database

-f The request originates from a DHCP server

-h Client host name (value is substituted by the DHCP server)

-d Client dynamic domain name (value is substituted by the DHCP server)

-s Indicates that the command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The following command string is appropriate for most cases. The string contains:

d;a;*; Delete all A (host name-to-IP-address) records for this host.

a;a;%s;
Add an A record using an IP address (%s) provided by the server, where %s indicates string substitution.

s;%s; Send a lease time (%s) (value provided by the server for the IP address), where %s indicates string substitution.

3110400;
The effect of this value is to reserve the host name for an additional time interval after the A record expiration. In this case, an interval of 36 days (3110400 seconds) is added to the A record expiration time. This value works to preserve a user's host name during times when the name-to-address mapping might expire, such as holidays, vacation, or other times of inactivity.

q Quit delimiter for the command string

For more information on client control of A record updates for non-dynamic DHCP clients, see client option81 in Specifying DHCP Options.

Releasing a Client A Record

Use the releaseDNSA keyword at a global level as a template which tells the DHCP server how to release a client record, when the client requests release. The template is adequate for normal purposes, but may be modified if necessary.

The keyword is:

```
releaseDNSA string
```

The following is an example of a releaseDNSA string issued:

```
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q""
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database

-f The request originates from a DHCP server

-h%s
Client host name (value is substituted by the DHCP server)

-s Information in the command string releases the DHCP client's A record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;a;%s;
Delete the A (host name-to-IP-address) record for this host.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0; An additional time interval added to the normal expiration of the host

name beyond the expiration of the A resource record. In this case, the value is zero because the A resource record is deleted.

q Quit delimiter for the command string

Specifying the Keyword for PTR Record Updates

DHCP servers ″own″ the PTR records and can update DDNS servers with a PTR record (resource records that map an IP address to a host name) for each address allocated to a host that identifies itself with DHCP options 12 (host name) and 15 (domain name), or with option 81, Client DHCP-DNS.

To enable DHCP server PTR record updates, use:

```
updateDNSP command_string
```

The following is a typical updateDNSP string issued by the DHCP server:

```
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q""
```

This default string is normally adequate. Typical changes to the command string are to modify an the name expiration extension time such as 0. This example instructs the DDNS name server to delete all PTR records for this address and add a record that maps the IP address to the new host name for the specified lease time. The updateDNSP command is issued when the DHCP client is assigned an address.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database

-f The request originates from a DHCP server

-r%s
The IP address which is to be mapped to the new host name

-s The command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The command string is appropriate for most cases. The string contains:

d;ptr;*;
Delete all PTR (IP-address-to-host name) records for this IP address.

a;ptr;%s;
Add a PTR record which maps the IP address to the host name (%s) provided by the server.

s;%s; Send a lease time (%s) (value provided by the server).

0 The effect of this value is to reserve the IP address entry this many seconds beyond when the IPaddress-to-host name mapping expires. In this case, no additional time is added.

q Quit delimiter for the command string

Note: The updateDNSP keyword is equivalent to the updateDNS keyword in earlier releases. The updateDNS keyword continues to be supported.

For more information on how a non-dynamic IP client updates its A and PTR records, see *DHCP and Dynamic IP Introduction*. For more information about the nsupdate command and its other parameters, see *DNS Administration*.

Releasing a Client PTR Record

Use the releaseDNSP keyword at a global level as a template which tells the DHCP server how to release the client PTR record, when the client requests release. The template is adequate for normal purposes, but may be modified if necessary. For example, this DHCP server keyword enables immediate release of a PTR record when a mobile client's lease is terminated.

The DHCP server command is:

```
releaseDNSP string
```

The following is an example of a releaseDNSP string:

```
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q""
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database

-f The request originates from a DHCP server

-r%s
The IP address substituted by the DHCP server

-s Information in this command string releases the DHCP client's PTR (IP-address-to-host-name mapping) record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;ptr;%s;
Delete the PTR record for this address.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0 An additional time interval added to the normal expiration of the PTR resource record beyond the expiration of the PTR resource record. In this case, the value is zero because the record is deleted.

q Quit delimiter for the command string

Defining DHCP Proxy Authority

Defining DDNS support also includes ensuring the DHCP domain is recognized at the DDNS server and setting up the necessary security for making updates to the DDNS server. To enable A record and PTR record dynamic update support in your DHCP server:

- Ensure that the administrator of the DDNS server configures dynamic reverse domains (*.in-addr.arpa) for all addresses for which the DHCP server will be making dynamic PTR updates.
- Create key file entries in the DDNS.DAT file for each dynamic reverse domain to be updated. If you use DDNS in your network, you must also create a DDNS.DAT file, which contains the security keys used to process IP address and host name updates with a DDNS server. For more information about the DDNS server, see DNS Administration.

Defining DDNS Key Files for the PTR Record

The DDNS.DAT file contains the security key pairs to be used for processing IP address-to-host name and host name-to-IP-address updates for the specified

addresses which are sent to the corresponding primary DDNS server. The file must contain at least one entry per primary DDNS server.

Run the nsupdate command on the server to create key entries to the /etc/ddns.dat file for DHCP supported clients. The format of the nsupdate command for creating security key pairs for updating PTR records in the DDNS.DAT file is:

```
nsupdate -f -hinverse_ip_address -g -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server

-h The inverse IP address(es) for which keys are generated.

inverse_ip_address
The format of the inverse_ip_address is the IP address in reverse with in-addr.arpa appended. For example, 9.67.149.111 as an inverse_ip_address entry is 111.149.67.9.in-addr.arpa. You can use a wild card to expand the scope of the entry. For example, if you want this entry to apply to all hosts whose IP address begins with 9.67, you could specify *.67.9.in-addr.arpa. If you want this entry to apply to all hosts, you could specify *.in-addr.arpa.

-g A key pair should be created for this entry.

-p The primary DDNS server

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.33.37.9.in-addr.arpa -p ns-updates.company.com
```

The wild card (*) allows aggregate entries such as *.33.37.9.in-addr.arpa to represent all addresses beginning with the three octets 9.37.33.

The format of the entry in the DDNS.DAT file for a DHCP server is:

```
inverse_ip_address  primary_ddns_server  private_key  public_key
```

The inverse_ip_address and primary_ddns_server are explained above.

The private_key and public_key are used to provide fail-save authentication for updates submitted by this DHCP server. As stored in the DDNS.DAT file, the private key is encrypted.

Defining DDNS Key Files for the A Record

The format of the nsupdate command for creating security key pairs for updating A records in the DDNS.DAT file is:

```
nsupdate -f -g -hfully_qualified_host_name -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server

-h The host name(s) for which keys are generated.

fully_qualified_host_name
The format of the fully_qualified_host_name is the host name followed by the full-qualified domain name or a wild card to allow aggregate entries.

-g A key pair should be created for this entry.

-p The primary DDNS server

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.city.company.com -p ns-updates.company.com
```

You can specify a specific host, such as myhost, or use a wild card to allow aggregate entries such as *.city.company.com to represent all hosts in the zone.

Configuring the BINL Server

The BINL server (binlsd) is similar to the DHCP server and is started and configured similarly. Port 4011 must be reserved for the BINL server. To reserve port 4011 for binlsd, add the following line to the PORT statement of the TCPIP profile:

```
4011 UDP BINLSD
```

Also, if the system is running in a multiple stack (common INET) environment, the INADDRANYPORT range cannot include 4011.

Following are the command line options that can be used with binlsd:

Option Description

-? Display the help message.

-b Display the program banner.

-q Execute in quiet mode.

-v Execute in verbose mode.

-f Override default configuration file location.

The BINL configuration file, dhcpsd.cfg by default, is searched for first in the current working directory. If not found in the current working directory, the /etc directory is searched. Following is a sample BINL configuration file:

```
######################################################################
#
#   binlsd.cfg -- BINL (Image Server) Configuration File
#
#   This file contains the directives that can be specified by the
#   server's administrator to configure the server and enforce
#   policies. This file is only a sample.  The finished file must be
#   placed in the directory specified by the ETC environment variable.
#
#   Do not put any long line without spaces in this file.
#
#   A line starting with a '#' character is a comment and is ignored.
#   A '#' on a line which is not part of a quoted string indicates
#   that anything to the right of this character is a comment and should
#   be ignored.
#
#   A continuation character of '\' is supported.  It must be
#   the last non-whitespace character on the line prior to any comments.
#
#   The directives are specified in the form of
#   <keyword><value1>...<valueN>.
#
```

```
#   Here is a partial list of keywords whose value can be specified
#   in this file:
#
#   Keyword             Effect
#   -------------       ----------------------------------------------------------------
#   sa                  Specifies the system architecture.
#   nit                 Specifies the network interface type.
#   lsa1nic             Specifies the lsa1 nic type.
#   tftp                Specifies the address of the tftpd server.
#   bpname              Specifies the install filename given to clients.
#   numLogFiles         The number of log files desired.
#   logFileSize         The size of log files in kilobytes.
#   logFileName         The name of the most recent log file.
#   logItem             An item to be logged.
#   servername          The LCM server name for LSA-1 clients. Not used for LSA-2 clients.
#   serverdomainname    The LCM domain name for LSA-1 clients. Not used for LSA-2 clients.
#
#   client              Definition of a set of options for a specific client
#                       or a definition of a client not to be serviced.
#
#   pxevendor           Configuration delimiter to indicate pxe options to follow.
#
#   pxeoption           A pxe configuration option value to pass to clients.
#
#   The scope of a keyword is limited by a pair of curly brackets ({, })
#   within which the keyword is located.  If a keyword is located outside
#   of any pair of curly brackets, its scope is applicable to all the
#   clients served by this server.  The curly brackets must appear alone on
#   a line.
#
#  Log files.  This set of parameters specifies the log files that will be
#  maintained by this server.  Each parameter is identified by a keyword
#  and followed by its value.
#
#  Keyword      Value           Definition
#  --------     ------------    ------------------------------------------
#  numLogFiles  0 to 99         number of log files.  If 0 is specified,
#                               no log file will be maintained and no log
#                               message is displayed anywhere.  When a log
#                               file reaches maximum size, a new log file
#                               is created, until the maximum number of
#                               log files have been created.  Only the most
#                               recent n log files are kept/
#
#  logFileSize  in K bytes      maximum size of a log file.  When the size
#                               of the most recent log file reaches this
#                               value, it is renamed and a new log file is
#                               created.
#
#  logFileName  file path       name of the most recent log file.  Less
#                               recent log files have the number 1 to
#                               n-1 appended to their names; the larger
#                               the number, the less recent the file.
#
#  logItem                      An item that will be logged.
#               SYSERR          System error, at the interface to the platform.
#               OBJERR          Object error, in between objects in the process.
#               PROTERR         Protocol error, between client and server.
#               WARNING         Warning, worth of attention from the user.
#               EVENT           Event occurred to the process.
#               ACTION          Action taken by the process.
#               INFO            Information that might be useful.
#               ACNTING         Accounting information on clients served.
#               TRACE           Code flow, for debugging.
#
#
```

```
#  servername   <servername>
#                                If present this name is sent to LSA-1 clients in the offer
#                                packet.  If this option in not present the server will send
#                                its name to the client as the LCM server.
#  serverdomainname <domainname> <workgroup|domain>
#                                Specifies the domain name of the LCM server that is sent to
#                                LSA-1 clients in the offer packet.
#
#                                <domainname> the name of the domain
#
#                                The last parameter can be the string "workgroup" to indicate
#                                that <domainname> is a workgroup or the string "domain" to
#                                indicate that <domainname> is a domain.

#  client <id_type><id_value><exclude|value>
#                                Definition of a client record.
#
#                                <id_type> is one of the hardware types defined
#                                in RFC 1340 (e.g. 1 for 10 megabit Ethernet,
#                                6 for 802.5 Token Ring.)  The type may be
#                                0, in which case the hardware type is not
#                                specified and the
#                                id_value may be a string of any format.
#
#                                <id_value> is a character string if the type
#                                is 0. Typically, this would be a domain name.
#                                For a non-zero <id_type>, the <id_value> is
#                                a hexadecimal string representing
#                                the hardware address of the client.
#
#                                The last paramter can be blank to indicate that
#                                the client matching <id_type> and <id_value>
#                                should get the specific parameters defined in
#                                its scope.
#
#                                The last parameter can be the string "exclude" to
#                                indicate that the client matching
#                                <id_type> and <id_value> should
#                                not be serviced by this server.
#
#       The client statement may be immediately followed
#       by a pair of curly brackets, in which the options
#       particular to this client can be specified.
#
#       Note: All clients inherit all globally defined options.
#
#
#
#  Pxeoption.  This keyword identifies an option statement.  The options assigned
#  are defined in the "Wired for Management Baseline Version 1.1" specification
#  authored by the Intel Corporation.
#
#  An option is specified by the "pxeoption" keyword followed by the option code
#  of this option and its data field, in a single line.  One or more options
#  may be specified.
#
#  The scope within which an option applies is delimited by a pair of curly
#  brackets ({, }) surrounding the option statement.
#
#  If two or more options with the same option code are specified, the
#  one with the most specific scope is used.  This allows, for example,
#  an option specified at the sa and nit scope to override that same option
#  specified at the global scope.  If two or more options
#  with the same option code are specified within the same scope,
#  the first one read by the server will be the one used, (subject to
#  its being overridden by the same option in a more specific scope).
#
```

```
#  All options specified will be sent to the client encapsulated in DHCP option 43.
#

#  Sa.  This parameter specifies the system architecture.  At the global level sa is
#  considered to be "unspecified" and any sa which is received will match if a more
#  specific sa is NOT explicitly configured.  A specific sa may be configured and values
#  for boot path name, TFTP server, and options maybe scoped under this specific
#  specification.
#
#  NIT. This parameter specifies the network interface type. Works in a similiar manner as
#  the system architecture (Sa.)  It can be configured to more specifically qualify a SA
#  (e.g sa 0 nit 1) or if configured alone will be considered to more specifically qualify
#  the unspecified SA.
#
#  LSA1NIC.  This parameter specifies the network interface type for LSA-1 clients.  Works
#  in a similiar manner as the system architecture (Sa.) and network interface type (NIT.)
#  Specific lsa-1 nic types to provide values for boot path name and TFTP server.  No
#  additional options are sent to the LSA-1 client.
#
#
#  TFTP.   This parameter specifies the address of the tftp server that contains
#  the install image.
#
#  Keyword             Value                 Definition
#  -----------   ----------------------    -------------------------
#  tftp          [ipaddress|hostname]       If "ipaddress" actual address of image server.
#
#                                             If "hostname" dns lookup is done to resolve
#                                               ipaddress of image server.
#
#
#  bpname.  The fully qualified pathname of the install image file.
#
#  Keyword           Value
#  -----------     -------------------
#  bpname         [filename]
#
#
###############################################################################

# The remaining portion of this file is an sample configuration file.
# Comments are added to assist in understanding the configuration file.
# Further information and detail is found in the online user documentation.

# Setup of the log file information.  This includes the size and name of the
# logfile along with number of logfiles maintained and type of information that
# will be logged.
numLogFiles     10
logFileSize     1000
logFileName     binlsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE

# default TFTP server
tftp   9.33.44.55

# Fully qualified boot Image pathnmae
bpname c:\tftp\lccm\lccm.1

# Global Options
```

```
pxevendor
{
  pxeoption 2 555
  pxeoption 3 544
  pxeoption 4 5
  pxeoption 5 5
}
#Excluded client list
client 1 080aab4567 exclude
client 1 000abc45d7 exclude
client 6 080aab4899 exclude

sa 0 nit 1
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1

  #Options specific to this sa and nit values
  pxevendor
  {
    pxeoption 1 224.1.1.1
    pxeoption 2 1758
    pxeoption 3 1759
    pxeoption 4 1
    pxeoption 5 2
  }
}

nit 2
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1

  #Client excluded from service if it falls into this category.
  client 6 08aa343bf3 exclude

  # Special configuration for client following into this nit2 type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nit2\nt40s
  }
}

lsa1nic 53
{
  tftp 9.180.71.79
  bpname  d:\lcm\system\images\53.X

  # Special configuration for client following into this nic type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nic2\nt40s
  }

  # Deny this client LSA1 service
  client 6 08005aab6abc4 exclude

}

#
# end of binlsd.cfg
#
```

Configuring a DDNS Server

There is no explicit configuration utility for the DDNS server as there is for the DHCP server. You can either create new DDNS server configuration files, or you can migrate an existing DNS configuration to dynamic DNS server configuration files.

There are three ways to use the DDNS server:

- Static DDNS server
- Dynamic secure DDNS server
- Dynamic pre-secured DDNS server

When used as a static DDNS server, there is nothing you have to do but use your existing DNS configuration files with the DDNS server. It will then work exactly the same way as the previous DNS server.

When used in dynamic secure mode, the DDNS server will allow clients to update their resource records dynamically using encryption keys that have been created by the clients themselves.

When used in dynamic pre-secured mode, the DDNS server will only allow those clients to update their records to which an encryption key has been provided that has been generated by the administrator.

Creating a New DDNS Server Configuration

The files required for a minimum configuration are:

- The nameserver boot file contains the path and file names for any other configuration files. The default for this file is /etc/named.boot. It will be examined by the DDNS server at startup.
- The nameserver domain file contains information about the zones for which this server will be authoritative, and all mappings from names to IP addresses (ordinary or forward name resolution). The file name is defined in the nameserver boot file.
- The nameserver reverse file contains information about the mappings from IP addresses to names (inverse or reverse name resolution). The file name is defined in the nameserver boot file.

Use the following steps to create DDNS server configuration files from scratch:

1. Create the DDNS configuration files. You can also modify the samples that are shipped in the /usr/lpp/tcpip/samples directory.

   A nameserver boot file might look as follows:

```
;
; boot file for name server configuration.
;
directory /test/dns/zones/
;
; type       domain                      source file or host
;
primary  test.itsc.raleigh.ibm.com        test.data       dynamic secure
;
primary  200.200.200.in-addr.arpa        db.200.200.200  dynamic secure
;
```

On the primary statements, you can specify if you want to use the DDNS server in dynamic secure or in dynamic presecured mode by using either the *dynamic secure* or the *dynamic presecure* keywords. A nameserver domain file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
@   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
             ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

A nameserver reverse file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

2. After you have created the files, use the NSUPDATE -g command to create the encryption key pairs for the zone resource records in the domain and reverse files.

   After the administrator has copied the public key into the files, they might look as follows:

   - Domain Name file:

     ```
     ;
     ;*******************************
     ;*  Start of Authority Records  *
     ;*******************************
     ;
     @   IN  KEY     80 0 1 AQPOzUYWvAUyZhYxogDcrtxOZOH33V31Tmrs1Db1WYiyI4Y
                            7Mmoz6Vm3XY/QTMHOyeHcVAMKmuba+rW4/+IkMeP3
     @   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
     ```

```
             ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

- Reverse Name file:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  KEY     80 0 1 AQPR+3ObXCgcjmBfKSnN4fD6v
                                              VH/AUIwincGNeD1MAuz2BTQSQ
                                              /bJkXLA3nxfV+HxKfxWptkRck
                                              wzxEk1DD3DSB
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

The NSUPDATE -g command will also create the DDNS.DAT file that contains the private encryption keys to sign any updates to the zone resource records in the domain and reverse files. This is shown in the following example:

```
test.itsc.raleigh.ibm.com ns-updates.test.itsc.raleigh.ibm.com
Pb7bySIfzXcW...

200.200.200.in-addr.arpa ns-updates.test.itsc.raleigh.ibm.com
KlexSRMP/q/k...
```

3. Start the DDNS server.
4. If you have a DHCP server configured for DDNS updates, you need to add an entry into the DDNS.DAT file using NSUPDATE -g that represents a 'wildcard' entry for the zones that DHCP will update (*.test.itsc.raleigh.ibm.com).

Note: KEY and SIG resource records, as well as encryption keys, always use a single line. The examples were indented for illustration purposes only.

Configuring for Presecured Mode Operation

IBM Open Edition's Dynamic DNS server supports two modes of securing updates to a dynamic DNS zone. In the default mode, called **dynamic secured**, any host that complies with the DDNS protocol may create resource records in a zone declared as dynamic. Once created, these records may only be updated by the administrator or by the host that created them.

In presecured mode, only hosts that have been pre-authorized by a DDNS administrator may create resources in a particular dynamic zone. Once created, these records may only be updated by the administrator or by the host that has been pre-authorized.

Functionally, the difference in these modes is whether or not the DDNS server will allow the creation of a KEY RR or whether it must already have a KEY RR defined for a resource in the zone. In the case of presecured mode, the KEY RR must be already defined in the zone before an update is accepted. Specifically, the administrator must enter the KEY RR data in the domain file for each client that will be making updates.

To configure a particular DNS dynamic zone for presecured mode operation, you must:

1. Specify **dynamic presecured** on the primary zone statement in the DDNS server boot file and create the corresponding domain file.
2. Run the nsupdate -g command to create an /etc/ddns.dat with the zone key information. Create a zone KEY RR.
3. Start the DDNS server by entering **named** at an OS/390 UNIX command prompt.
4. For each client host:
   a. Use **nsupdate** with the -g parameter, which will generate a key for the host and save it in /etc/ddns.dat.
   b. Use **nsupdate** with the -a parameter to dynamically register and save a host's KEY RR in the DDNS server domain file.
   c. Manually extract the /etc/ddns.dat file key entry for the host into a separate DDNS.DAT file and distribute it to the end user for installation into their /etc subdirectory.

The following example is a scenario demonstrating an administrator using **nsupdate** to pre-register an end user in a **dynamic presecured** domain. Administrator input is highlighted in bold:

```
[C:\]nsupdate -g -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
---
Key Gen ...... succeeded ...


[C:\]nsupdate -a -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> a
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): key
DDNSUpdate_KEY (Add Flags 0000 Protocol 0 Algid 1
      Keylen 64 Key10-15Õ: AQO+w3f2H1F0zM2Q ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> s
```

```
..sig Expiration (secs from now, ENTER for 3600):
..sig KEY pad (ENTER for default of 3110400):
DDNSSignUpdate ...succeeded
DDNSFinalizeUpdate ...succeeded
DDNSSendUpdate ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> q

[C:\]
```

Control Entries, Resource Records, and Special Characters

You use control entries, resource records and special characters to create the domain data files (forward and reverse) and the loopback file.

Control Entries

The Standard Resource Record Format defines two control entries:

$ORIGIN
Specifies a domain name to be appended to each host name that does not end with a dot. In the following example, the origin, `raleigh.ibm.com`, is appended to the host name, `host1`:

```
$ORIGIN raleigh.ibm.com.
host1   A     1.2.3.4
```

Note: The domain specified applies to the host names that follow the $ORIGIN entry, until another $ORIGIN entry resets the origin to another domain name. The origin for host names that precede an $ORIGIN statement is the domain specified by the primary directive in the boot file.

$INCLUDE
Specifies a file to be included in the current file, for example:

```
$INCLUDE /etc/named.inc
```

The origin for any host names in the included file that do *not* end with a dot is the same as the origin specified in the current file.

Resource Records

The resource record format is the basis for all entries in the hints, loopback, and domain data files. This section describes each type of record.

Note: This section is not intended to be a comprehensive description of resource records. For complete information, read the appropriate RFCs that are available from the InterNIC. (For instructions on obtaining RFCs, see “Recommended Reading” on page 712.)

The general format of a resource record is:

{name} {ttl} {address_class} record_type record_data

name
Specifies the name of the zone or host system associated with a record. This field is optional. If the field is blank, the name server uses the name of the zone

or host system from the preceding resource record. You can use the special character @ in place of a zone name if the zone name is the same as the zone defined in the boot file.

ttl Specifies the time-to-live value, in number of seconds, which is the amount of time that this record is valid in a cache. This field is optional. The default is the time-to-live value contained in the minimum field in the file's SOA record.

address_class
Specifies the address class for this entry. The allowable values are described below. This field is optional; the default is IN.

HS Specifies HESIOD class.

IN Specifies the TCP/IP-based Internet. You can also use the full name of the address class, `INTERNET`.

record_type record_data
Specifies the type of record and associated data. The following record types and record data are valid.

A *IP_address*
Indicates an address record, which contains the dotted-decimal IP address for the name identifying the record. This record provides forward mapping (host name-to-IP address). For host systems with multiple addresses, use one A record per address.

AFSDB *subtype server_host_name*
Indicates an Andrew File System Data Base location. The AFSDB record allows mapping from a domain name to the name of an AFS cell data base server. This record contains the domain name of a host that has a server for the cell name indicated in the name field of the resource record. The subtype value indicates the type of service.

CNAME *canonical_name*
Indicates a canonical name record, which provides the alias or alternative name for a host system. Enter the alias in the *name* field (for example, `elmer CNAME abc.xyz`, where `elmer` is the alias and `abc.xyz` is the name of the host system). When the name server searches for the alias, it finds the CNAME record, substitutes the canonical name for the alias, and then searches for the A record identified by the canonical name.

HINFO *processor operating_system*
Indicates a host-information record, which specifies the names of the processor type and operating system of a host system. Enclose names in quotation marks (for example, `mvs004 IN HINFO "3090" "OS/390"`). The HINFO record is typically used only for administrative purposes and is accessed by tools that query the database, such as `onslookup`.

ISDN *ISDN_address subaddress*
Indicates the ISDN address of the owner, whose name is specified in the *name* field, as well as any direct-dial-in number.

KEY *flags protocol algorithm public_key*
This record represents a public encryption key for a name in the DNS database. This can be a key for a zone, a host, or a user. A KEY RR is authenticated by a SIG RR. The *flags* field indicates the type of resource record for which this KEY RR is provided. The *protocol* field indicates the protocols (in addition to DDNS) that are to be secured for authentication by this KEY RR. The *algorithm* field indicates what encryption algorithm should be used with this key; in case of IBM Dynamic IP this field has a value of 1,

which means that the RSA/MD5 algorithm is being used. The *public_key* field is the actual public key to be used for authentication. This field is structured in a public exponent length field, the public key exponent portion, and the public key modulus portion. See RFC 2065 for more details on KEY RR formats.

MX *preference exchanger*
Indicates a mail exchanger record, which identifies a host capable of acting as a mail exchanger for the domain specified in the name field. The MX record type is followed by the mail preference, which is the priority number used to rank the mail exchangers. Mailers attempt delivery first to the mail exchangers with the lowest preference value. If delivery fails, the host with the next highest value is tried. The highest possible preference value is zero; the next highest preference value is 1, and so on. Hosts with identical preference values are selected randomly. Create an MX record for every host that receives mail.

NS *name_server*
Indicates a name server record, which contains the name of an authoritative name server (primary or secondary) for the domain specified in the name field. The NS record type is followed by the name of the name server.

NSAP *record_length record_data*
Indicates name-to-Network Service Access Points (NSAP) mapping. The value in the record_length field is an unsigned 16-bit integer specifying the number of octets in the record_data field. The data value is the encoded binary value of the NSAP as it is formatted in the CLNP (Connectionless Network Protocol) source or destination address field.

PTR *host_name*
Indicates a pointer record, which is used in reverse domain data files (the in-addr.arpa domain). This record contains the host name referenced by an IP address. The IP address is listed in reverse octet order, concatenated with the in-addr.arpa string. For example, if a host named Host1 has an IP address of 9.67.43.100, the address in the reverse domain data file as 100.43.67.9.in-addr.arpa.

PX *preference RFC822_address X.400_address*
Indicates a PX pointer to X.400/RFC822 mapping information.

RT *preference intermediate_host_name target_host_name*
Provides route-through binding for hosts that do not have their own direct wide-area-network addresses. This record contains a route preference and the names of the intermediate and target hosts.

RP *mbox host_name*
Indicates a responsible person record, which specifies the person or group responsible for a particular zone or host system. Each host_name parameter must have an associated TXT resource record.

SIG *type_covered algorithm labels original_TTL signature_expiration*

time_signed key_footprint signer's_name signature
This record represents a digital signature that authenticates a set of resource records in the database. The *type_covered* field indicates the type of RR covered by this signature. The *algorithm* field indicates what encryption algorithm should be used with this key; in case of IBM Dynamic IP this field has a value of 1, which means that the RSA/MD5 algorithm is being used. The *labels* field indicates the number of labels (host and domain name strings separated by dots) in the SIG owner name. The

original_TTL field is the original time to live for the signed resource record. It is included to prevent caching nameservers from decrementing this value. It is protected by the signature, and it is different from the TTL of the SIG record itself. The *signature_expiration* is the time at which the signature becomes not valid. This value is represented in a number of seconds starting from January 1, 1970, GMT (ignoring leap seconds). The *time_signed* is the time when this signature has actually been signed, represented in the same format as the *signature_expiration*. The *key_footprint* field determines, depending on the applicable encryption algorithm, how to decode the signature. The *signer's_name* is the fully qualified domain name of the signer generating this SIG RR. The *signature* is the actual digital signature that authenticates a set of RRs of the type indicated in the *type_covered* field. See RFC 2065 for more details on SIG RR formats.

SOA *authoritative_name_server resp_person (serial refresh retry expire minimum)***)**

Indicates the start of authority record, which specifies that the current name server is authoritative for the zone. The SOA record also specifies values that are used primarily by secondary name servers.

- *authoritative_name_server*— The name of the name server authoritative for the zone.
- *resp_person*— The mail address of the individual or group responsible for maintaining the zone data. This is the mail address to contact for registering children name servers in the Domain Name System.

You can add or use parentheses to group the following parameters. See "Special Characters" on page 791.

- *serial*— The serial number of the domain database, which identifies the current version of the data and which is referenced by secondary name servers. Increment the number every time you change the file. One possible notation is YYYYMMDD.
- *refresh*— The refresh interval, which indicates the length of time (in seconds) a secondary server for this domain allows between transfers from the primary name server.
- *retry*— The retry interval, which indicates the length of time (in seconds) a secondary server for this domain should allow before retrying a failed transfer.
- *expire*— The expiration time, which indicates the length of time a secondary server should consider its data valid in the event it is not able to contact the primary name server. If there is no contact with the primary name server prior to the expiration time, the secondary server considers its data stale, and no longer responds to queries for its domain.
- *minimum*— The default minimum time-to-live (ttl) value, which is attached to all data given in response to a query or a secondary transfer and which determines the length of time the data can be cached. This value is only used to supply a default value if a resource record does not provide a ttl.

TXT *text_data*

Indicates a text string record, which contains descriptive text. The TXT keyword is followed by any descriptive text enclosed in quotation marks. The TXT record is typically used only by tools that query the database, such as `onslookup`.

The TXT record is also used to specify a *secure zone*. A secure zone is accessible only by certain networks or hosts. To implement secure zones, you must configure the named daemon with a secure zone defined in at least one TXT resource record.

The format for a secure zone TXT record is:

secure_zone {addr_class} TXT *string*

The syntax for the string is either ″host IP address:H,″ which allows queries from the specified host or ″network address:netmask,″ which allows queries from the specified network. (To use the default netmask for the network, omit the netmask from the string.) Include the loopback address in the secure zone record to enable name resolution by local clients.

WKS *address protocol (service_list)*
Indicates a well-known services record, which describes the services provided by a host on a protocol basis. Each defined TCP/IP service has a unique protocol number; see RFC 1060 for more information. The WKS keyword is followed by the IP address, a protocol number, and a list of the services provided.

You can use parentheses with the WKS resource record. See “Special Characters”.

X25 *PSDN_address*
Indicates a public switched-data-network address for the name.

Special Characters

The following characters have special meanings:

@ Indicates the current domain (origin).

. Indicates the null (root) domain.

Note: You can also use a trailing dot (period) after any name in a file to prevent the current origin from being appended. For example, if you enter the host name `elmer.raleigh.ibm.com`, add a dot after `com` to prevent the name server from re-appending the origin.

**** When followed by any character other than a digit, denotes that the character, rather than the character’s special meaning, is to be used. For example, in a mail box specification, you can use a backslash followed by a dot (\.) to place a dot in the local part of the name. The name server treats the dot as a normal character and not as the end of the name.

( ) Specifies the grouping of data that spans lines. The ends of lines are not recognized within parentheses.

Note: SOA and WKS resource records are the only resource records that support parentheses.

; Specifies a comment, which can appear anywhere on a line. The remainder of the line is ignored by the name server.

Boot File Directives

A boot file is organized in lines, using the following directives:

bogusns *IP_address(es)*
Directs the name server not to query other name servers that give incorrect information. The IP address or addresses of the bogus name servers are listed in the second field.

cache *. {file_path} file_name*
Identifies the location of the hints (root server) file. The hints file contains the names of the root servers. The dot (period) in the second field indicates the root domain. You do not need to specify the file path if you specify the working directory first, using the directory directive.

directory *file_path*
Specifies the working directory for the boot file. Specifying a working directory lets you use relative file paths for the other files in the boot file. Use only one directory directive in a boot file and list it before any other directives that specify file names. If you do not use the directory directive, the name server defaults to the `/etc` directory.

forwarders *IP_address(es)*
Specifies the IP addresses of servers that can accept unresolved queries. The forwarders are queried in the order presented in the list until one is successfully contacted.

include *file_name*
Specifies that the file named in the second field is to be included in the boot file.

limit transfers-in *number of transfers*
Limits the number of zone transfer processes allowed at any one time.

limit transfers-per-ns *number of transfers*
Limits the number of zone transfers a secondary name server can request of a given name server at any one time. The default is two.

options fake-iquery
Directs the name server to respond to inverse queries with false information rather than with an error message. Use this directive only if you cannot prevent clients from sending inverse queries. An inverse query provides resource record data and requests the names and IP addresses of all hosts containing that data.

options forward-only
Directs the name server to send queries only to the forwarders specified by the forwarders directive. The options forward-only directive is equivalent to the slave directive, and must be accompanied by the forwarders entry in a boot file.

options no-recursion
Prevents the name server from responding to recursive queries.

options query-log
Directs the name server to log all queries with the syslog daemon. You can use the SIGWINCH signal to toggle query logging on and off.

primary *domain_name file_name* **{***cluster***} | {dynamic | dynamic secure |**

dynamic presecure} {nokeytosec}
Identifies a zone for which this server is the primary and the file from which to

load its data. *cluster* is not allowed with dynamic, dynamic secure, or dynamic presecure. nokeytosec is valid only for dynamic zones.

cluster Sysplex connection-balanced zone. *cluster* is not allowed with dynamic, dynamic secure, dynamic presecure, or nokeytosec.

dynamic
See dynamic secure.

dynamic secure
Automatic registration by the user.

dynamic presecure
Must be administered.

nokeytosec
Indicates that this server will not send KEY or SIG to a secondary server on a zone transfer request. Nokeytosec is only valid for dynamic zones.

Note: The primary record in the boot file of the name server for a connection-optimized sysplex may contain an additional parameter, `cluster`. See "Step 5: Configure the Sysplex Name Servers" on page 734.

secondary *domain_name IP_address(es) {backup_file_name} {cluster}*
Identifies the domain for which this server is a secondary name server, the IP address or addresses of the name server (typically the primary) from which it gets its data, and, optionally, the file from which it copies its data. Multiple addresses can be specified, in which case the secondary name server tries each address in the order listed, until the zone transfer is complete.

Note: The secondary record in the boot file of the name server for a connection-optimized sysplex may contain an additional parameter, `cluster`. See "Step 5: Configure the Sysplex Name Servers" on page 734.

slave
Performs the same function as the `options forward-only` directive.

sortlist *network(s) or IP_address(es)&{network mask}*
Specifies ″closeness″ for referring to other name servers and for sorting addresses sent in response. ″Closeness″ refers to the similarity of addresses in the network list. Requests for name resolution from a client on the same subnet as the server receive local network addresses first, addresses in the same network second, addresses in the sortlist third, and all remaining addresses last. You can list multiple sortlist entries.

Use subnet masks to identify subnetworks or specific hosts. There are no spaces between the ampersand character and the IP address or between the ampersand and the network mask.

stub *domain_name IP_address(es) file_name*
Used in boot files of primary name servers to identify the subdomains and their name servers. The domain name is the zone for which the server is a stub server. The IP address or addresses is the IP address or addresses of the subdomain's primary name server, and the file name is the name of the file where the forward data is stored.

Note: The stub directive indicates that the name server is not authoritative for the data in the given zone.

xfrnets *IP_address(es)&{network mask}*
Places limits on the secondary name servers that can receive data from a primary name server. The *IP addresses* are the IP addresses of the secondary name servers that can receive data.

Use subnet masks to identify subnetworks or specific hosts. Note that there are no spaces between the ampersand character and the IP address or between the ampersand and the network mask.

Sample Files

The following sections provide sample files. The notes after each sample explain some of the highlights.

Sample Forward Domain Data File

Following is the sample forward domain data file that is provided with the product. Refer to the program directory for its location. The file specifies the host systems in three networks (7.0.0.0, 8.0.0.0, and 9.0.0.0). The host system dog is a router between two of the networks.

```
;   /etc/named.for file
;
$ORIGIN ibm.com.
raleigh    IN  SOA  buzz1.raleigh.ibm.com. bug@vmail (                  {1}
   1       ; Serial—  change when database is changed
   10800   ; Refresh— secondary checks every three hours
   3600    ; Retry—   secondary retries connection
           ;            every hour after a failed zone transfer
   604800  ; Expire—  data expires after one week
   86400 ) ; Time to Live— data cached in other servers one day)
;
$ORIGIN raleigh.ibm.com.                                                 {2}
             IN      NS      buzz1                                       {3}
             IN      NS      elmer
             IN      NS      jumbo
; OWNER     CLASS   TYPE      RECORD DATA
localhost    IN      A       127.0.0.1
buzz1        IN      A       9.37.34.149
elmer        IN      A       9.37.34.7
jumbo        IN      A       8.37.34.12
dog          IN      A       8.37.34.113                                 {4}
             IN      A       9.37.34.113
vmail        IN      A       7.37.34.1
kent         IN      A       7.37.34.2
greg         IN      A       7.37.34.3
printserver  IN      A       7.37.34.4
             IN      A       8.37.34.33
             IN      A       9.37.34.44
gary         IN      CNAME   jumbo                                       {5}
```

{1} The SOA (Start of Authority) record specifies the name server buzz1 as the authoritative name server for the domain raleigh.ibm.com. The mail address of the person responsible for domain data is bug@vmail. The numbers enclosed in parentheses are parameters used primarily to manage the zone information.

{2} The control entry $ORIGIN appends the string raleigh.ibm.com. to all the following host names that do not end with a dot ('.').

{3} The NS (Name Server) records specify the name servers in the zone. Note that NS records do not distinguish between primary and secondary name servers.

{4} This A (Address) record maps the host name (dog.raleigh.ibm.com) to the IP addresses of the two networks to which it is connected.

{5} The CNAME record specifies that the name gary is an alias for the host name jumbo.raleigh.ibm.com.

Sample Reverse Domain Data File

The following is a sample reverse domain data file that maps IP addresses to host names in the network 9.37.0.0. The addresses of host systems in the other two networks (7.0.0.0 and 8.0.0.0) would be listed in separate reverse domain data files.

```
;    /etc/named.rev

$ORIGIN 37.9.in-addr.arpa.
34 IN  SOA  buzz1.raleigh.ibm.com. bug@raleigh.ibm.com. (
               1  10800  3600  604800  86400 )          {1}
               IN      NS      buzz1.raleigh.ibm.com.   {2}
               IN      NS      elmer.raleigh.ibm.com.
$ORIGIN 34.37.9.in-addr.arpa.                           {3}
7              IN      PTR     elmer.raleigh.ibm.com.   {4}
149            IN      PTR     buzz1.raleigh.ibm.com.
113            IN      PTR     dog.raleigh.ibm.com.
44             IN      PTR     printserver.raleigh.ibm.com.
```

{1} These lines are the start of authority (SOA) entry, which lists information about the domain, including the authoritative name server. Like a forward domain data file, a reverse domain data file can have only one SOA resource record. The left and right parentheses mark the beginning and end, respectively, of the parameters used to manage zone information.

{2} These NS (Name Server) records specify two name servers, buzz1 and elmer, as the authoritative name servers in the zone.

{3} The $ORIGIN control entry appends the string 34.37.9.in-addr.arpa. to all addresses not ending with a dot.

{4} This pointer (PTR) entry maps the IP address 7.34.37.9.in-addr.arpa. to the host name elmer.raleigh.ibm.com.

Sample Hints (Root Server) File

The hints file contains the names and addresses of name servers in the root domain.

```
;   This file holds the information on root name servers needed to
;   initialize cache of Internet domain name servers
;   (e.g., reference this file in the "cache   .   <file>"
;   configuration file of BIND domain name servers).
;
;   This file is made available by InterNIC registration services
;   under anonymous FTP as
;   file                /domain/named.root
;   on server           FTP.RS.INTERNIC.NET
;   -OR- under Gopher** at RS.INTERNIC.NET
;       under menu      InterNIC Registration Services (NSI)
;         submenu       InterNIC Registration Archives
```

```
;       file               named.root
;
;   last update:       May 19, 1997
;   related version of root zone:      1997051700
;
;   formerly NS.INTERNIC.NET
;
.                       3600000   IN  NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.   3600000       A    198.41.0.4
;
;   formerly NS1.ISI.EDU
;
.                       3600000       NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.   3600000       A    128.9.0.107
;
;   formerly C.PSI.NET
;
.                       3600000       NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.   3600000       A    192.33.4.12
;
;   formerly TERP.UMD.EDU
;
.                       3600000       NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.   3600000       A    128.8.10.90
;
;   formerly NS.NASA.GOV
;
.                       3600000       NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.   3600000       A    192.203.230.10
;
;  formerly. NS.ISC.ORG
;
.                       3600000       NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.   3600000       A    192.5.5.241
;
;   formerly NS.NIC.DDN.MIL
;
.                       3600000       NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.   3600000       A    192.112.36.4
;
;   formerly AOS.ARL.ARMY.MIL
;
.                       3600000       NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.   3600000       A    128.63.2.53
;
;   formerly NIC.NORDU.NET
;
.                       3600000       NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.   3600000       A    192.36.148.17
;
;   temporarily housed at NSI (InterNIC)
;
.                       3600000       NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.   3600000       A    198.41.0.10
;
;   housed in LINX, operated by RIPE NCC
;
.                       3600000       NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.   3600000       A    193.0.14.129
;
;   temporarily housed at ISI (IANA)
;
.                       3600000       NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.   3600000       A    198.32.64.12
;
;   temporarily housed at ISI (IANA)
```

```
;
.                       3600000       NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000       A    198.32.65.12
; End of File
```

Sample Loopback File

Loopback addresses let hosts and clients direct communications to themselves. The following is a sample loopback file used for all three name servers in the raleigh.ibm.com domain (buzz1, elmer, and jumbo).

```
;    /etc/named.lbk

0.0.127.in-addr.arpa. IN SOA   buzz1.raleigh.ibm.com. bug@vmail (   {1}
     1
     10800
     3600
     604800
     86400    )
0.0.127.in-addr.arpa.     IN  NS   buzz1.raleigh.ibm.com.    {2}
0.0.127.in-addr.arpa.     IN  NS   elmer.raleigh.ibm.com.
0.0.127.in-addr.arpa.     IN  NS   jumbo.raleigh.ibm.com.

1.0.0.127.in-addr.arpa.   IN PTR   localhost.                {3}
```

{1} These records are the SOA (Start of Authority) entry, which lists information about the zone. The left and right parentheses mark the beginning and end, respectively, of the parameters used to manage zone information.

{2} These three records identify the three name servers in the zone.

{3} The PTR (Pointer) record indicates the common loopback address of the host systems.

Sample Boot File

A boot file directs a name server to its data files. The sample boot file below is provide with the product. Refer to the program directory for its location.

```
;   /etc/named.boot
;
; type           domain              source file or host
directory     /etc/dnsdata                          {1}
primary       raleigh.ibm.com           named.for   {2}
primary       34.37.9.in-addr.arpa      named.rev   {3}
primary       0.0.127.in-addr.arpa      named.lbk   {4}
cache      .                            named.ca    {5}
options query-log                                   {6}
```

{1} The directory directive specifies `/etc/dnsdata` as the working directory for the boot file and eliminates the need for a file path for the data files, loopback file, and hints file.

{2} This primary directive specifies the server on which this boot file is installed as the primary server for the domain raleigh.ibm.com. The directive also specifies that the name-to-address mappings are in the forward domain data file, `named.for`.

{3} This primary directive specifies the server on which this boot file is installed as the primary server for the reverse domain 34.37.9.in-addr.arpa and specifies that the address-to-name mappings are in the reverse domain data file, `named.rev`.

{4} This primary directive specifies the loopback network and file for the host.

{5} The cache directive specifies the hints file, which lists the top-level domain name servers. The file path and name of the hints file is `/etc/named.ca`.

{6} This directive tells the name server to log all the queries it receives to the syslog log file.

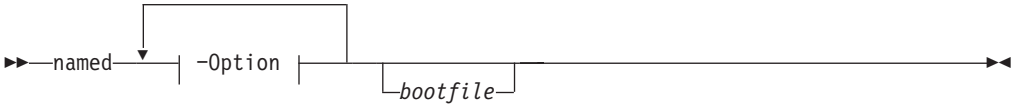
The following is an example of a caching-only name server boot file:

```
directory /usr/local/named   ; or your data directory
primary 0.0.127.in-addr.arpa  db.127.0.0    ; for loopback address
cache   .                     db.cache
```

The named Daemon

Use the named daemon to start the name server. The named daemon runs on name server hosts and controls the name resolution function.

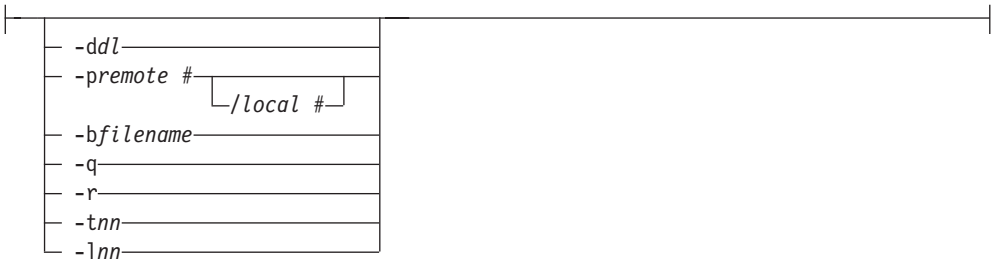
Syntax



Usage Notes

The configuration options of the named daemon specify the tasks you can perform with the daemon. If you do not specify the `-b` option or `bootfile`, the named daemon reads the default boot file `/etc/named.boot`.

Option:



-d *dl*
Specifies a debugging option and causes the named daemon to write debugging information to the file `/tmp/named.run`. Debug information generated during zone transfers is written to `/tmp/xfer.ddt.XXXXXX`, where XXXXXX is a unique identifier. Note that one of these files will be generated for each zone for which the named daemon is a secondary server.

If the debug level (*dl*) variable is six or greater, then a trace of the resources exchanged during the last initiated zone transfer is written to `/tmp/xfer.trace`. Valid debug levels are one to 11, where 11 supplies the most information.

If named is started from the OS/390 shell with the `-d` option, use the `&` shell operator at the end of the command line to run named asynchronously. If you do not, the named tracing process occupies the OS/390 shell.

-p *remote # {/local #}*
Reassigns the port that is used in queries to other name servers. (The default is 53.) The remote option specifies the port the name server uses to query other name servers. The local option specifies the port that named binds to. The remote port number is the same as the local port number if the local port number is not specified.

-b *filename*
Specifies an alternate boot file to `/etc/named.boot`.

-q Enables the logging of queries received by the name server. The queries are written to the syslog file by the syslog daemon.

-r Disables recursive query processing.

The following options apply only to connection optimization in a sysplex domain. For a complete discussion of connection optimization, see “Connection Optimization in a Sysplex Domain” on page 723.

-t *nn*
Specifies the time (nn, in seconds) between refreshes of sysplex names and addresses and of the weights associated with those names and addresses. The default is sixty seconds.

-l *nn*
Specifies the time-to-live (nn, in seconds) for sysplex names and addresses after they are sent into the network. The default is zero seconds.

Modifying

Six name server signals are available for working with the named daemon:

SIGHUP
Reloads the boot file, `named.boot`, from the disk.

SIGINT
Dumps the name server’s database and hints file into the `/tmp/named_dump.db` file.

SIGABRT
Dumps the current statistics of the name server in the `/tmp/named.stats` file.

SIGUSR1
Starts debug tracing for the name server and causes the named daemon to write debugging information to the file `/tmp/named.run`. Debug information generated during zone transfers is written to `/tmp/xfer.ddt.XXXXXX`, where XXXXXX is a unique identifier. Note that one of these files will be generated for each zone for which the named daemon is a secondary server.

If the debug level (*dl*) variable is six or greater, then a trace of the resources exchanged during the last initiated zone transfer is written to `/tmp/xfer.trace`. Valid debug levels are one to 11, where 11 supplies the most information.

SIGUSR2
Stops debug tracing.

SIGWINCH
Toggles query logging on and off.

A sample MVS start procedure is included in the samples directory that lets you issue these signals to the name server from the MVS operators console. The name of the sample is *nssig*. It has one parameter, `sig`. Values for the `sig` parameter are the same values that are valid for the `-s` parameter of the OS/390 UNIX kill command applicable to the name server (HUP, INT, ABRT, USR1, USR2, and WINCH). A typical invocation from the MVS operators console would look like the following if the sample procedure were unaltered:

```
s nssig,sig=hup
```

Examples

- To start the named daemon from the OS/390 shell, enter the following command:

  ```
  named
  ```

  To start the named daemon from the MVS operators console, enter the following command:

  ```
  s named
  ```

  where `named` is the name of your name server start procedure.
- To stop the named daemon from the OS/390 shell, enter the following command:

  ```
  kill -9 $(cat /etc/named.pid)
  ```

  The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup. To stop the named daemon from the MVS operators console, display a list of tasks and stop the running named job. The job is typically called `named1`.
- To get short status from the named daemon, follow the steps below:
  1. Enter the following command from the OS/390 shell:

     ```
     kill -ABRT $(cat /etc/named.pid)
     ```

     The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup.
  2. Check the file `/tmp/named.stats`.
- To enable debug message logging for the named daemon, enter the following command from the OS/390 shell:

  ```
  kill -USR1 ($cat /etc/named.pid)
  ```

  The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup.

  Each time this command is issued, it causes the named daemon to increment the debug level and write debugging information to the file `/tmp/named.run`. Debug information generated during zone transfers is written to `/tmp/xfer.ddt.XXXXXX`, where XXXXXX is a unique identifier. Note that one of these files will be generated for each zone for which the named daemon is a secondary server.

  If the debug level (*dl*) variable is six or greater, then a trace of the resources exchanged during the last initiated zone transfer is written to `/tmp/xfer.trace`. Valid debug levels are one to 11, where 11 supplies the most information.
- To disable debugging for the named daemon, enter the following command from the OS/390 shell:

  ```
  kill  -USR2   $(cat /etc/named.pid)
  ```
- To turn query logging on, enter the following command from the OS/390 shell:

  ```
  kill  -WINCH  $(cat /etc/named.pid)
  ```

  The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup. To turn off query logging, send another `WINCH` signal to the name server.

  Use query logging to identify resolver configuration errors. When query logging is turned on, a running name server logs every query with syslog. The displayed

syslog messages include the IP address of the host that made the query and the query itself. Before logging queries, make sure that the syslog daemon is logging `LOG_INFO` messages.

Note: You can also turn query logging on by inserting the directive `options query-log` in the name server boot file or by starting the name server with `-q` on the command line.

- To dump the contents of the name server database, follow the steps below:
  1. Enter the following command from the OS/390 shell:

     ```
     kill  -INT   $(cat /etc/named.pid)
     ```

     The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup.
  2. Check the file `/tmp/named_dump.db`.
- To reload the `named.boot` boot file from the disk, enter the following command from the OS/390 shell:

  ```
  kill  -s HUP $(cat /etc/named.pid)
  ```

  The process ID of the named daemon is stored in the `/etc/named.pid` file upon startup.

onslookup/nslookup—Querying A Name Server in Command Mode

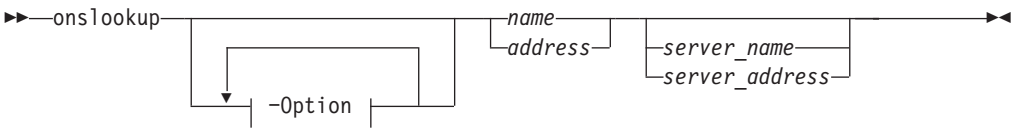
Use the onslookup command to specify a single query.

Note: The OS/390 UNIX shell provides a synonym (nslookup) for onslookup. For more information about this synonym, see *OS/390 SecureWay Communications Server: IP User's Guide*.

To display a list of options, enter the following from the command line:

```
onslookup -?
```

Syntax



Parameters

name
Queries the name server for the current query-type of name. The name typically represents a host name.

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

server_name
Directs the default name server to map *server_name* to an IP address and then use the name server at that address. This argument is optional. The default is the default name server found by the search order described in "onslookup/nslookup Configuration" on page 719.

server_address
Specifies the IP address of the name server to be queried other than the default name server. A query for the address in the in-addr.arpa domain is initially made to the default name server to map the IP address to a domain name for the server. This argument is optional. The default is the default name server found by the search order described in "onslookup/nslookup Configuration" on page 719.

Usage Notes

Parameter values and domain names are not case sensitive.

To display a list of options, enter the following from the command line:

```
onslookup -?
```

onslookup messages are not documented in the CS for OS/390 library. Therefore, onslookup command messages do not give a message ID for debugging.

Related Topics

For a complete list and description of onslookup options, see “onslookup/nslookup Options” on page 807. For a selection of sample queries using onslookup options, see “Examples” on page 811.

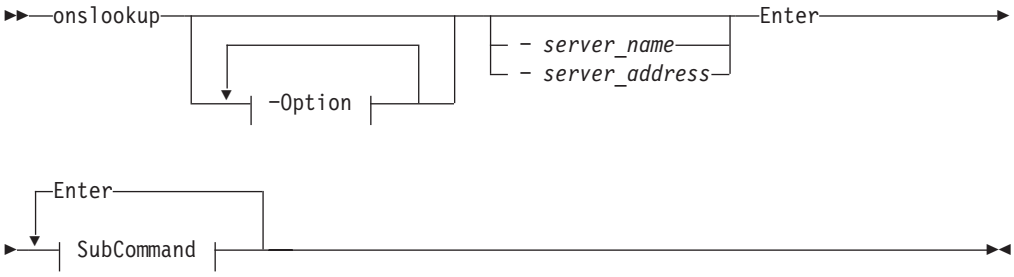
onslookup/nslookup—Issuing Multiple Queries in Interactive Mode

Use the interactive mode to issue multiple queries. In interactive mode, an initial query is made to the selected name server to verify that the server is accessible. All subsequent interactive queries are sent to that server unless you specify another server using the `server` or `lserver` subcommands.

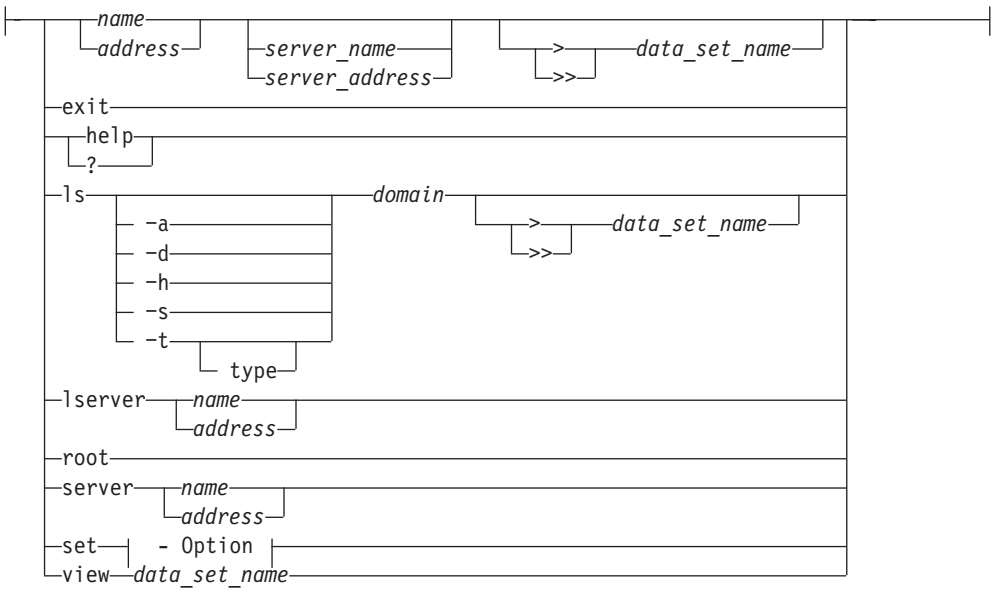
Note: The OS/390 UNIX shell provides a synonym (nslookup) for onslookup. For more information about this synonym, see *OS/390 SecureWay Communications Server: IP User's Guide*.

The command line length must be less than 256 characters. To treat a built-in command as a host name, precede it with an escape character (\). An unrecognized command is interpreted as a host name.

Syntax



SubCommand:



Parameters

Query overview
Perform a query for the name nominated. (The name is typically a host name.)

The query requests all information about the name using the current class and query (resource record) type. You can specify a server other than the current server to perform name resolution.

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

The use of *address* is only intended for A and PTR query types. If the current query type is address (A) or domain-name pointer (PTR), onslookup generates a PTR type query for the specified address in the `in-addr.arpa` domain. This returns PTR records, which define the host name for the specified address. If the current query type is neither of these two types, a query is performed using the current query type, with the domain name specified as the address given.

Text that does not conform to the defined options and follows the preceding syntax is treated as a domain query. onslookup does not issue a query for a domain name if the name is unqualified and is the same as one of the defined options unless the name is preceded by the escape character.

exit
Exits from onslookup interactive mode.

help or ?
Displays a brief summary of commands.

ls *parms*
Lists the information available for the domain. By default, the IP address of each node in the domain is listed.

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

-a CNAME aliases and canonical names

-d ALL

-h HINFO

-s WKS

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

See “Resource Records” on page 787 for information about valid query types.

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

Output can be placed in a data set for later viewing by specifying *data_set_name* for a specified domain. The > *data_set_name* option places the output in *data_set_name* and overwrites the contents, if any, of the data set.

The >> *data_set_name* option places the output in *data_set_name* and appends it to the contents, if any, of the data set. There must be at least 1 space before and after the > or >> symbol.

lserver *parms*
Changes the current server using the *original* name server to find the new name server. The parameter *name* or *address* is required.

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

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

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

server *parms*
Changes the current server using the *current* name server to find the new name server. The parameter *name* or *address* is required.

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

set *option*
Changes internal state information values. See “onslookup/nslookup Options” for a description of the options.

view *data_set_name*
Sorts and lists the contents of *data_set_name* one screen at a time. An error occurs if the data set does not exist.

onslookup/nslookup Options

The configuration options of onslookup determine the operation and results of name server queries. These options can be specified in command-mode queries, interactive-mode queries, or by the methods described in “onslookup/nslookup Configuration” on page 719. In particular, see Table 27 on page 720.

When you include onslookup options with the initial onslookup command, the - operand must immediately precede the option. If you specify onslookup options while in interactive mode, the `set` subcommand must precede the option.

For example, to specify a name server (NS) type record lookup for the domain name `fourex.oz` in command mode you enter:

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

To submit the same request using interactive mode you enter the following sequence:

```
onslookup
enter
set querytype=ns
enter
fourex.oz
```

The − operand is not valid preceding *options* in the `.onslookuprc` file. To make *querytype of NS* a default option for your onslookup commands place the following statement in the `.onslookuprc` file:

```
set querytype=ns
```

or

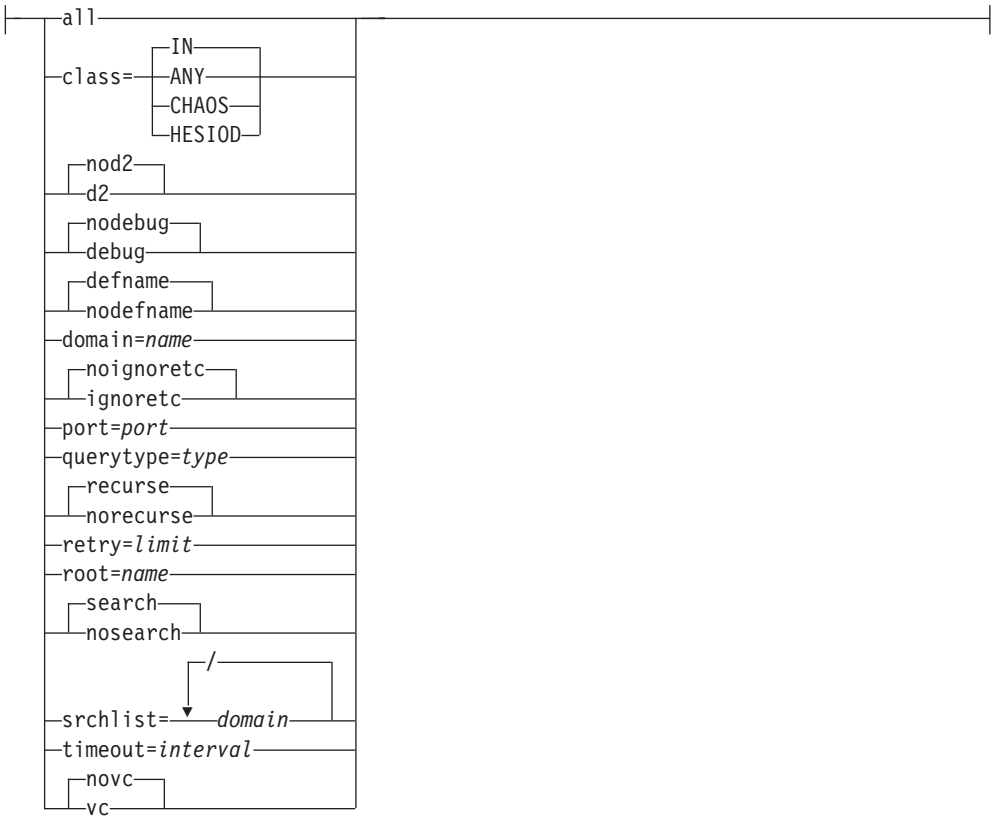
```
querytype=ns
```

The optional `.onslookuprc` file contains only onslookup options and defines the onslookup defaults. If the `.onslookuprc` file exists, the onslookup options are read from the file and executed before any queries are made. You must enter each option on a separate line. Blank lines are ignored.

The following is an example of the contents of the `.onslookuprc` file:

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

Option:



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

class=*class*
Sets the class of information returned by queries. The minimum abbreviation for this option is `cl`. The default is IN (Internet).

d2 Directs onslookup to enable extra debugging mode. Using `d2` also enables debug mode.

Note: Use this option to view all the information returned by the name server.

nod2
Directs onslookup to disable extra debugging mode. The default is `nod2`.

debug
Directs onslookup to print debugging information for and the response to each query. The minimum abbreviation is `deb`.

nodebug
Directs onslookup not to print debugging information for each query and its corresponding response. This option also disables `d2`. The minimum abbreviation is `nodeb`. This is the default.

defname
Directs onslookup to append the default domain name to an unqualified domain name in a query. The minimum abbreviation for this option is `def`.

You can change the default domain name using the `domain=`*`name`* option.

nodefname
Directs onslookup not to append the default domain name to an unqualified domain name in a query.

If you specify this option, the domain name specified in the query is passed to the server without modification. The minimum abbreviation for this option is `nodef`.

domain=*name*
Sets the default domain name to *name*. The validity of *name* is not verified. This option also updates the search list to contain the specified domain and the two higher-level domains (parent and grandparent) of the default domain if it has at least two components. For example, if the default domain is

```
mugwump.wurrup.fourex.oz
```

the search list will contain

```
mugwump.wurrup.fourex.oz
wurrup.fourex.oz
```

and

```
fourex.oz
```

Use the *set srchlist* subcommand and option to specify a different list. The minimum abbreviation for the `domain` option is `do`.

ignoretc
Directs onslookup on the handling of truncated responses. When you specify this option, the name server indicates, in the response header, that the complete query response did not fit into a single UDP packet and has been truncated. The minimum abbreviation for this option is `ig`.

Specifying `ignoretc` directs onslookup to ignore the truncation when it is set in the response by the name server.

onslookup does not handle responses greater than 512 characters in length. Responses greater than 512 characters are truncated and the internal truncation flag is set. This condition is revealed only when the debug option is enabled.

noignorectc
Directs onslookup to automatically retry the query using a TCP connection when a response is sent with the truncation indicator set. This is the default. The minimum abbreviation for this option is `noig`.

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

querytype=*type*
Specifies the type of information returned by queries. The default query type is `A` (address information). Using the wild card value `ANY` specifies a global query that returns all resource records for a specific domain name. The minimum abbreviation for this option is `q`.

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

recurse
Directs onslookup to request a recursive query when querying a name server. The minimum abbreviation for this option is `rec`. This is the default.

norecurse
Specifies that a recursive query is not returned. The minimum abbreviation for this option is `norec`.

retry=*limit*
Specifies the number of times a request is sent. When a request is sent and the time-out period expires for a response, the request is resent until the value specified in *limit* has been reached. The value specified in *limit* determines the total number of attempts made to contact the name server. The default value for *limit* is retrieved from the resolver configuration file. The minimum abbreviation for this option is `ret`.

Attempting to set *retry* to zero results in a retry value of one. One is the minimum value allowed.

The retry algorithm for onslookup uses both the *limit* value and the time-out period. Each time a request is resent, the time-out period for the request is twice the time-out period used for the last attempt.

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

search
Directs onslookup to enable the use of a search list. The minimum abbreviation for this option is `sea`. This is the default.

nosearch
Directs onslookup not to use a search list. The minimum abbreviation for this option is `nosea`.

srchlist=*[domain|domain|...]*
Specifies zero to six domain names to be appended to unqualified host names when attempting to resolve a host name. Each domain name is tried in turn until a match is found.

This option also sets the default domain to the first domain name specified in the search list. The minimum abbreviation for this option is `srchl`.

timeout=*interval*
Specifies the number of seconds to wait before a request times out. The minimum abbreviation for this option is `t`. The minimum value is 1.

vc Directs onslookup to use a virtual circuit (TCP connection) to transport queries to the name server or datagrams.

novc
Specifies to not use a virtual circuit to transport queries to the name server or datagrams. This option is the default.

Examples

This section contains examples of the onslookup command mode queries and interactive mode queries using the various options available for onslookup commands.

In Figure 35 on page 812, a network is described by a single zone in the domain name hierarchy stored in the name servers. The nameserver, MVS152, is the primary nameserver for the zone. The letters *TR* stand for Token Ring, and *U41* represents a 3172 interconnect controller. The queries are answered using the name server on the `MVS152` host in the network.

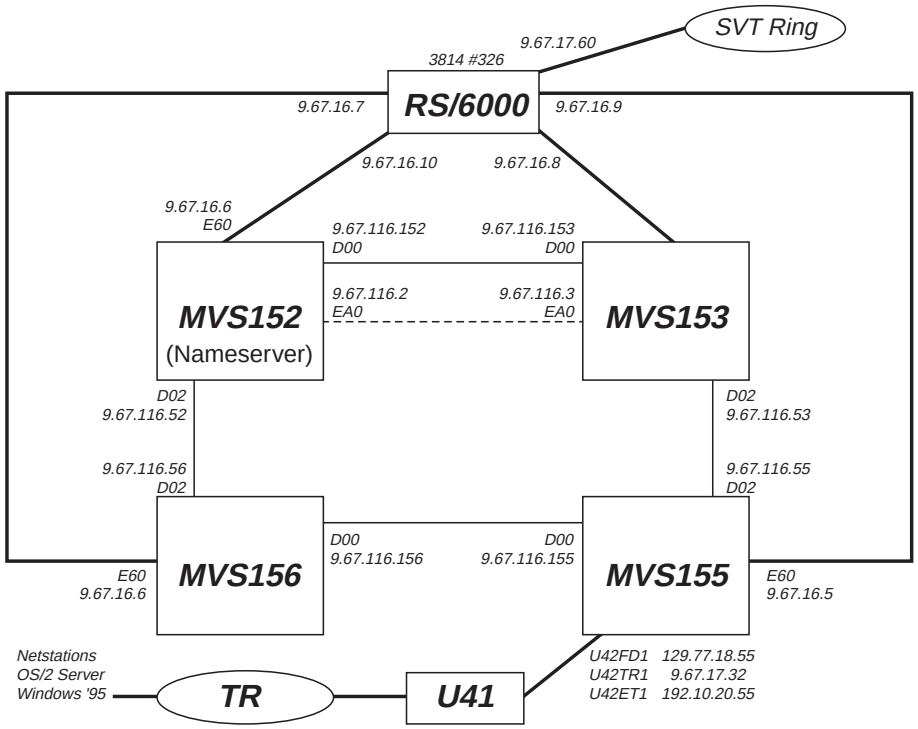


Figure 35. Sample Network

The following examples are command mode queries.

1. To make a simple address query:

```
> mvs152.tcpmvs.tcp33.raleigh.ibm.com
Server:  mvs152.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.152

Name:    mvs152.tcpmvs.tcp33.raleigh.ibm.com
Addresses:  9.67.116.2, 9.67.116.52, 9.67.116.252, 9.67.116.152,  9.67.16.6
```

2. To specify a name server (NS) type record lookup:

```
> set type=ns
> tcpmvs.tcp33.raleigh.ibm.com
Server:  mvs152.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.152

tcpmvs.tcp33.raleigh.ibm.com    nameserver =mvs153.tcpmvs.tcp33.raleigh.ibm.com

tcpmvs.tcp33.raleigh.ibm.com    nameserver =mvsesa07.tcp.raleigh.ibm.com
tcpmvs.tcp33.raleigh.ibm.com    nameserver =mvs152.tcpmvs.tcp33.raleigh.ibm.com

mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.16.4
mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.53
mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.33
mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.23
mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.153
mvs153.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.3
mvs152.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.152
mvs152.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.16.6
mvs152.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.2
mvs152.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.52
mvs152.tcpmvs.tcp33.raleigh.ibm.com     internet address = 9.67.116.252
```

The following command places onslookup in interactive mode with `MVS153` as the default server.

```
> onslookup - mvs153.tcpmvs.tcp33.raleigh.ibm.com
  Default Server:  mvs153.tcpmvs.tcp33.raleigh.ibm.com
  Address:  9.67.116.3
```

All following examples are in the interactive mode initiated in the preceding example.

1. Show the default flag settings:

```
> set all
Default Server:  mvs153.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.3

Set options:
no debug        defname         search          recurse
no d2           no vc           no ignoretc     port=53
querytype=A     class=IN        timeout=5       retry=4
root=a.root-servers.net.
domain=tcpmvs.tcp33.raleigh.ibm.com
srchlist=tcpmvs.tcp33.raleigh.ibm.com
```

2. Perform a simple address query:

```
> mvs153
Server:  mvs153.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.3

Name:    mvs153.tcpmvs.tcp33.raleigh.ibm.com
Addresses:  9.67.116.23, 9.67.16.4, 9.67.116.53, 9.67.116.33
          9.67.116.3, 9.67.116.153
```

3. Set the query record type to HINFO, and perform another query:

```
> set type=HINFO
> mvs152.sysplex.tcp33.raleigh.ibm.com
Server:  mvs153.sysplex.tcp33.raleigh.ibm.com
Address:  9.67.116.23

mvs152.sysplex.tcp33.raleigh.ibm.com   CPU = 3090    OS =  OS/390R4
sysplex.tcp33.raleigh.ibm.com nameserver=mvs152.sysplex.tcp33.
                                        raleigh.ibm.com
sysplex.tcp33.raleigh.ibm.com nameserver=mvs153.sysplex.tcp33.
                                        raleigh.ibm.com
sysplex.tcp33.raleigh.ibm.com nameserver=mvsesa07.tcp.
                                        raleigh.ibm.com
mvs152.sysplex.tcp33.raleigh.ibm.com internet address=9.67.116.152
mvs152.sysplex.tcp33.raleigh.ibm.com internet address=9.67.16.6
mvs152.sysplex.tcp33.raleigh.ibm.com internet address=9.67.116.2
mvs152.sysplex.tcp33.raleigh.ibm.com internet address=9.67.116.52
mvs152.sysplex.tcp33.raleigh.ibm.com internet address=9.67.116.252
mvs153.sysplex.tcp33.raleigh.ibm.com internet address=9.67.116.23
```

4. Find out the name servers available for a domain:

```
> set type=ns
> tcpmvs.tcp33.raleigh.ibm.com
Server:  mvs153.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.3

tcpmvs.tcp33.raleigh.ibm.com   nameserver =mvs153.tcpmvs.tcp33.raleigh.ibm
tcpmvs.tcp33.raleigh.ibm.com   nameserver =mvsesa07.tcp.raleigh.ibm.com
tcpmvs.tcp33.raleigh.ibm.com   nameserver =mvs152.tcpmvs.tcp33.raleigh.ibm.com
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.153
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.3
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.16.4
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.53
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.33
mvs153.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.23
mvs152.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.152
mvs152.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.16.6
mvs152.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.2
mvs152.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.52
mvs152.tcpmvs.tcp33.raleigh.ibm.com    internet address = 9.67.116.252
```

5. Change the server from MVS153 to MVS152 and make more queries:

```
> server mvs152
Default Server:  mvs152.tcpmvs.tcp33.raleigh.ibm.com
Addresses:  9.67.116.23, 9.67.16.4, 9.67.116.53, 9.67.116.33
          9.67.116.3, 9.67.116.153

> set type=a
> holly

> holly.tcpmvs.tcp33.raleigh.ibm.com
server:  mvs152.tcpmvs.tcp33.raleigh.ibm.com
address:  9.67.116.33

Name:    holly.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.130.204
```

6. Set debug, query, and disable debug:

```
> set deb
> mvs156
Server:  mvs153.tcpmvs.tcp33.raleigh.ibm.com
Address:  9.67.116.33

;; res_mkquery(0, mvs156.tcpmvs.tcp33.raleigh.ibm.com, 1, 1)
------------
Got answer:
 HEADER:
   opcode = QUERY  , id = 1491  , rcode = NOERROR
   header flags:   response  , auth. answer  , want recursion  , recursion

avail.
       questions = 1  , answers = 4  , authority records = 2  , additional
  QUESTIONS:
    mvs156.tcpmvs.tcp33.raleigh.ibm.com  , type = A  , class = IN
  ANSWERS:
->  mvs156.tcpmvs.tcp33.raleigh.ibm.com
    canonical name =mvs156.tcp33.raleigh.ibm.com
    ttl = 86400 (1 day)
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.116.156
    ttl = 120 (2 mins)
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.116.56
    ttl = 120 (2 mins)
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.16.3
    ttl = 120 (2 mins)
  AUTHORITY RECORDS:
->  tcp33.raleigh.ibm.com
    nameserver =mvs156.tcp33.raleigh.ibm.com
    ttl = 120 (2 mins)
->  tcp33.raleigh.ibm.com
    nameserver =mvs152.tcpmvs.tcp33.raleigh.ibm.com
    ttl = 120 (2 mins)
  ADDITIONAL RECORDS:
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.116.156
    ttl = 120 (2 mins)
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.116.56
    ttl = 120 (2 mins)
->  mvs156.tcp33.raleigh.ibm.com
    internet address = 9.67.16.3
    ttl = 120 (2 mins)
->  mvs152.tcpmvs.tcp33.raleigh.ibm.com
    internet address = 9.67.116.52
->  mvs152.tcpmvs.tcp33.raleigh.ibm.com
    internet address = 9.67.116.252
    ttl = 86400 (1 day)
->  mvs152.tcpmvs.tcp33.raleigh.ibm.com
    internet address = 9.67.116.152
    ttl = 86400 (1 day)
->  mvs152.tcpmvs.tcp33.raleigh.ibm.com
    internet address = 9.67.16.6
    ttl = 86400 (1 day)
->  mvs152.tcpmvs.tcp33.raleigh.ibm.com
    internet address = 9.67.116.2
    ttl = 86400 (1 day)

------------
Name:    mvs156.tcp33.raleigh.ibm.com
Addresses:  9.67.116.156, 9.67.116.56, 9.67.16.3
Aliases:  mvs156.tcpmvs.tcp33.raleigh.ibm.com

> set nodebug
```

7. Find all addresses in the domain using the `ls` option:

```
> ls tcpmvs.tcp33.raleigh.ibm.com
[mvs153.tcpmvs.tcp33.raleigh.ibm.com]
 tcpmvs.tcp33.raleigh.ibm.com.  server =mvs153.tcpmvs.tcp33.raleigh.ibm.com
 tcpmvs.tcp33.raleigh.ibm.com.  server =mvsesa07.tcp.raleigh.ibm.com
 tcpmvs.tcp33.raleigh.ibm.com.  server =mvs152.tcpmvs.tcp33.raleigh.ibm.com
 mvs153                         9.67.16.4
 mvs153                         9.67.116.53
 mvs153                         9.67.116.33
 mvs153                         9.67.116.23
 mvs153                         9.67.116.153
 mvs153                         9.67.116.3
 holly                          9.67.130.204
 mvs155                         9.67.116.55
 mvs155                         9.67.16.5
 mvs155                         9.67.17.32
 mvs155                         129.77.18.55
 mvs155                         192.10.20.55
 mvs155                         9.67.116.155
 mvsesa07                       9.67.116.207
 localhost                      127.0.0.1
 tony                           9.67.128.127
 mvs152                         9.67.116.52
 mvs152                         9.67.116.252
 mvs152                         9.67.116.152
 mvs152                         9.67.16.6
 mvs152                         9.67.116.2
```

8. Find all aliases (CNAME resource records) in the domain, then exit from onslookup interactive mode:

```
> ls -a tcpmvs.tcp33.raleigh.ibm.com
[mvs153.tcpmvs.tcp33.raleigh.ibm.com]
 hostc                          mvs155.tcpmvs.tcp33.raleigh.ibm.com
 host1                          mvs152.tcpmvs.tcp33.raleigh.ibm.com
 mvs156                         mvs156.tcp33.raleigh.ibm.com
 host152                        mvs152.tcpmvs.tcp33.raleigh.ibm.com
 host2                          mvs153.tcpmvs.tcp33.raleigh.ibm.com
 muckel                         tony.tcpmvs.tcp33.raleigh.ibm.com
 host153                        mvs153.tcpmvs.tcp33.raleigh.ibm.com
 host3                          mvs155.tcpmvs.tcp33.raleigh.ibm.com
 hec                            holly.tcpmvs.tcp33.raleigh.ibm.com
 mvs7                           mvsesa07.tcp.raleigh.ibm.com
 host155                        mvs155.tcpmvs.tcp33.raleigh.ibm.com
 hosta                          mvs152.tcpmvs.tcp33.raleigh.ibm.com
 corno                          holly.tcpmvs.tcp33.raleigh.ibm.com
 hostb                          mvs153.tcpmvs.tcp33.raleigh.ibm.com

> exit
```

9. To display a summary of available commands:

```
> help
Commands:  (identifiers are shown in uppercase, [] means optional)
NAME             -print info about the host/domain NAME using default server
NAME1 NAME2      -as above, but use NAME2 as server
exit             - exit the program
help or ?        - print info on common commands
ls [option] DOMAIN [>|>> FILE] - list addresses in DOMAIN(optional: output to FILE)
      -a           - list canonical names and aliases
      -h           - list HINFO (CPU type and operating system)
      -s           - list well-known services
      -d           - list all records
      -t TYPE      - list records of the given type (e.g.,  A,CNAME,MX, e
   lserver NAME     - set default server to NAME, using initial server
   server NAME      - set default server to NAME, using current default se
   root             - set current default server to the root
   view FILE        - sort an 'ls' output file and view it with more
 set OPTION       - set an option
      all          - print options, current server and host
      class=X      - set query class to one of IN(Internet),CHAOS,HESIOD or ANY
      domain=NAME  - set default domain name to NAME
      port=x       - use TCP/IP port number x
      querytype=X or type=x - set query type, e.g., A,ANY,CNAME,NS,PTR, etc
      retry=X      - set number of retries to X
      root=NAME    - set root server to NAME
      srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2,
      timeout=X    - set initial time-out interval to X seconds
      [no]debug    - print debugging information
      [no]d2       - print exhaustive debugging information
      [no]defname  - append domain name to each query
      [no]ignore   - ignore truncation errors
      [no]recurse  - ask for recursive answer to query
      [no]search   - use the search list
      [no]vc       - always use a virtual circuit
```

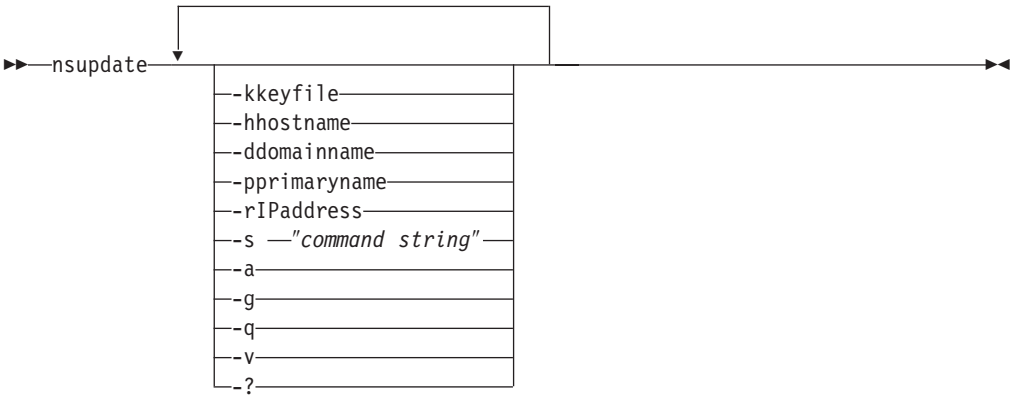
Usage Notes

- When specifying domain names that include dots, the trailing dot (indicating a fully qualified domain name) is optional. onslookup deletes the trailing period if it is present. If you are specifying a root domain, the domain name must have two trailing periods. For example, specify `mynode..` when the node `mynode` is in the root domain.
- A time-out error occurs if the name server is not running or is unreachable. A `Non-existent Domain` error occurs if any resource record type for the specified domain name is not available at the name server. A `Server Failed` error occurs when the local name server cannot communicate with the remote name server. See RFC 1034 for a complete description of name server return codes.
- onslookup does not recognize typing or syntax errors in subcommands. The query is still sent and the name server response printed. The response is usually `Non-existent Domain`, which indicates that the server could not find a match for the query.
- onslookup messages are not documented in the CS for OS/390 library. Therefore, onslookup command messages do not give a message ID for debugging.

nsupdate Command

You can use nsupdate in an interactive fashion where you are prompted through a series of subcommands and associated input values to create and execute DNS update operations on a host record. Alternatively, if you know the sequence of operations and input values beforehand, you can use nsupdate in ″batch″ mode and specify a subcommand sequence in the -s ″command string″ command line parameter.

Syntax



Parameters

-kkeyfile
By default, nsupdate uses /etc/ddns.dat for storing and retrieving key information. Use the -k parameter to specify an alternate key file.

-hhostname
Specifies the name or alias of a remote host.

-ddomainname
Specifies the name of the dynamic domain for the host. Not needed if the -h parameter specifies a fully-qualified host name.

-pprimaryname
Specifies the fully-qualified host name of the DDNS server that is primary.

-rIPaddress
Specifies the IP address used to update PTR records. (You cannot specify -r if you specify -h or -d.)

-s ″command string″
Command string is the string of subcommands with associated required input to be executed by nsupdate. Each subcommand is separated by a semi-colon.

The following illustrates how the -s parameter is used to add an A Record with an expiration time of one hour (3600 seconds), and a signature 36 days (3110400 seconds):

```
nsupdate -h warpspeed.dynozone.sandbox -s a;9.67.96.10;s;36
>
The above example assumes that the key file entry for this host already exists
in /etc/ddns.dat and contains the fully-qualified name of the
primary DDNS server.
```

-a The administrator mode flag. Specifies that the records in the update are to be signed and authenticated.

-g Only generates a key file entry for the host, but does not register the key with the DDNS server.

-q Specifies quiet mode. When this option is used, no prompting or informational messages are displayed.

-v Used for debugging purposes. When used on the command line, this turns on verbose mode. When verbose mode is on, all the requests to, and responses from, the name server are displayed.

-? Displays help for nsupdate.

nsupdate Subcommands

The following subcommands can be used in the **nsupdate** command shell.

- add
- new
- exists
- delete
- ttl
- sign
- quit

The **sign** subcommand signs and sends the transaction(s).

The **quit** subcommand quits the program. (Also Ctrl+C quits the program.)

The **ttl** subcommand sets the default TTL to be used in records on the update request. The default value is 4660.

The **delete** subcommand appends a DELETE.

The **new** subcommand appends an ADDNAMENEW.

The **exists** subcommand append an ADDNAMEEXIST.

The **add** command appends an ADD.

nsupdate Examples

The following are example console sessions using **nsupdate** in interactive mode. All examples assume the administrator has already set up zone key and that the private key component for the zone is included in the local /etc/ddns.dat.

How an Administrator Removes and Locks Out a Host Name

The following example demonstrates an administrator's input and the system responses when removing and locking out a host name. Administrator input is highlighted in bold:

1. Generate a New Key for the Host

```
[C:\]nsupdate -g -h warpspeed.dynozone.sandbox -p
netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
---
Key Gen ...... succeeded ...
```

2. Delete a User's A and KEY RRs and add a New KEY RR for New, Administrator Generated Key

```
[C:\]nsupdate -a -h warpspeed.dynozone.sandbox -p
netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> d
---
InitDDNSUpdate ...... succeeded ...
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): a
....ip addr: *
DDNSUpdate_A (Delete *) ...succeeded
```

```
Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> d
---
InitDDNSUpdate ...... succeeded ...
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): key
DDNSUpdate_KEY DELETE *
succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> a
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): key
DDNSUpdate_KEY (Add Flags 0000 Protocol 0 Algid 1
      Keylen 64 Key10-15Õ: AQPS80e7uGuuNIdA ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> s
..sig Expiration (secs from now, ENTER for 3600):
..sig KEY pad (ENTER for default of 3110400):
DDNSSignUpdate ...succeeded
DDNSFinalizeUpdate ...succeeded
DDNSSendUpdate ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> q

[C:\]
```

How an Administrator Creates an Alias for the Dynamic Zone

The following example demonstrates an administrator's input and the system responses when creating an alias for the dynamic zone. Administrator input is highlighted in bold:

```
[C:\]nsupdate -a -h ns-updates.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> a
---
InitDDNSUpdate ...... succeeded ...
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): cname
....hostname: netadmin
DDNSUpdate_CNAME (Add netadmin.dynozone.sandbox) ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> s
..sig Expiration (secs from now, ENTER for 3600):
..sig KEY pad (ENTER for default of 3110400):
DDNSSignUpdate ...succeeded
DDNSFinalizeUpdate ...succeeded
DDNSSendUpdate ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> q

[C:\]
```

Return Codes

Following are the return codes, origination of the return codes, and explanations for the most common problems that you might encounter:

Return Code	Origin	Explanation
0	N/A	Successful.

Return Code	Origin	Explanation
-2	Local error	Input error.
-10	Local error	No key found in ETC\DDNS.DAT. A key is needed because either -f was specified or there is a KEY RR already in the name server data.
-11	Local error	Invalid key in ETC\DDNS.DAT. Does not authenticate the user.
-12	Local error	No response received from the name server.
-1	Local error	Represents any other (local) error not specified above.
1	Server error	Format error. The name server was unable to interpret the request.
2	Server error	Server failure. The name server was unable to process this request because of a problem with the name server.
3	Server error	Name error. The domain name specified does not exist.
4	Server error	Not implemented. The name server does not support the specified Operation code.
5	Server error	Refused. The name server refuses to perform the specified operation for security or policy reasons.
6	Server error	Alias error. A domain name specified in an update is an alias.
7	Server error	Name Exists error. A name already exists. This return code is only meaningful from a server in response to an ADDNAMENEW operation.
8	Server error	Record error. Indicates that a resource record (RR) does not exist. This return code is only meaningful from a server in response to a DELETE operation.
9	Server error	Zone error. Indicates that the update is to be performed on a zone for which the server is not authoritative, or that the records to be updated exist in more than one zone.
10	Server error	Ordering error. If an ordering mechanism is used (for example, a SIG RR or a SOA RR), this code indicates an ordering error. Time-signed problems are also indicated by this return code.

Chapter 22. Configuring Simple Network Management Protocol (SNMP)

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the Simple Network Management Protocol (SNMP). The OS/390 UNIX osnmp command is the SNMP command used to access MIB object information from the OS/390 shell, as the NetView SNMP command does from NetView.

Processing SNMP Request

Figure 36 illustrates the interface between TCP/IP and the implementation of SNMP.

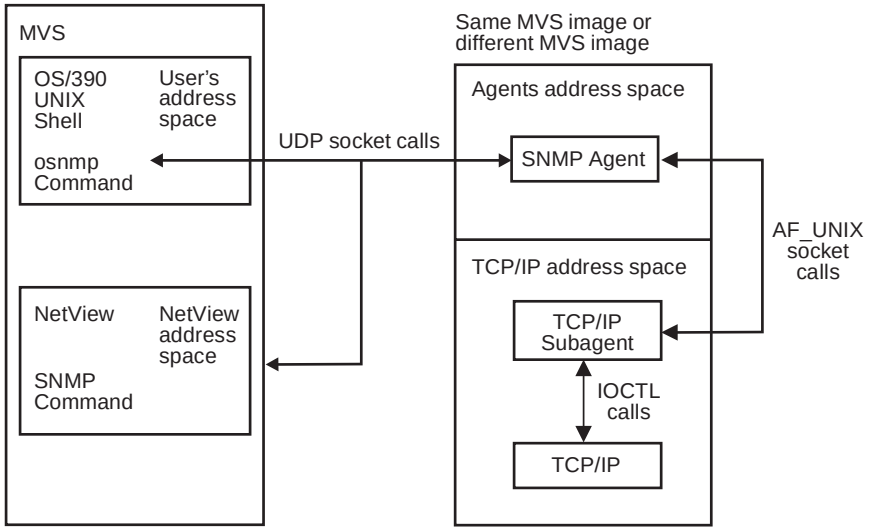


Figure 36. Overview of SNMP Support

This list illustrates the sequence of events from the time you issue an SNMP command until you receive the response:

1. The user issues a NetView SNMP (SNMP) or OS/390 UNIX SNMP (osnmp) command.
2. The command processor validates and encodes the request in a Protocol Data Unit (PDU), and sends it to the SNMP agent.
3. The SNMP agent validates the request and, if necessary, sends it to an SNMP subagent. Requests for agent-oriented objects are handled by the agent and all others are handled by a subagent. To determine which objects are handled by the agent and which by a subagent, refer to the *OS/390 SecureWay Communications Server: IP User's Guide*.
4. The agent sends the response to the originator of the request. The command processor displays the response.

Note: Although not shown in Figure 36 on page 823, other subagents, such as the OMPROUTE subagent shipped as part of CS for OS/390, also communicate with the SNMP agent using AF_UNIX socket calls or TCP socket calls from their own address spaces.

The SNMP agent and the SNMP subagents record trace information via the OS/390 UNIX syslog daemon using the *daemon* facility. For detailed information regarding syslogd and specifying the daemon facility in the /etc/syslog.conf configuration file, see "Appendix E. Syslog Daemon" on page 1183.

Deciding on SNMP Security Needs

The SNMP agent supports SNMPv1, SNMPv2c, and SNMPv3 security. SNMPv1 and SNMPv2c are community-based security, where a community name (or password) is passed with a request. If the community name is recognized as one that can be used by the IP address from which the request originates, the SNMP agent processes the request.

SNMPv3 provides a more powerful and flexible framework for message security and access control. Message security involves providing:

- Data integrity checking, to ensure that the data was not altered in transit
- Data origin verification, to ensure that the request or response originates from the source from which it claims to have come
- Message timeliness checking and, optionally, data confidentiality, to protect against eavesdropping

Access control is the ability to control exactly what data an individual user can read or write.

The SNMPv3 architecture introduces the User-Based Security Model (USM) for message security and the View-Based Access Control Model (VACM) for access control. The architecture supports the concurrent use of different security, access control, and message processing models. For example, community-based security can be used concurrently with USM.

USM uses the concept of a user for which security parameters (levels of security, authentication and privacy protocols, and keys) are configured at both the agent and the manager. Messages sent using USM are better protected than messages sent with community-based security, where passwords are sent in the clear and displayed in traces. With USM, messages exchanged between the manager and the agent have data integrity checking and data origin authentication. Message delays and message replays (beyond what happens normally due to a connection-less transport protocol) are protected against with the use of time indicators and request IDs. Data confidentiality, or encryption, is also available as a separately orderable product.

The use of VACM involves defining collections of data (called views), groups of users of the data, and access statements that define which views a particular group of users can use for reading, writing, or receipt in a trap.

SNMPv3 also introduces the ability to dynamically configure the SNMP agent using SNMP SET commands against the MIB objects that represent the agent's

configuration. This dynamic configuration support enables addition, deletion, and modification of configuration entries either locally or remotely. Remote modification of user keys can be especially useful.

Decide on your security needs—community-based or user-based.

If you are satisfied with the security of your existing configuration, you can continue to use community-based security with no migration. If you would like to take advantage of USM or VACM, you will need to migrate your configuration. Note that USM can be used only when both the SNMP agent and the manager requesting the data support USM, as the CS for OS/390 SNMP agent and the osnmp command do. VACM can be used even for community-based requests, but doing so requires migration of existing community name and trap destination definitions. Following is a list of the advantages and disadvantages of using each type of security.

Table 39. Security Advantages and Disadvantages

SNMPv1/SNMPv2c advantages	**SNMPv3 disadvantages**
Traditional standards-based administrative model	Emerging standards-based administrative model
Widely implemented on many platforms	Not yet implemented on many platforms
Easy to configure	More robust configuration options
SNMPv1/SNMPv2c disadvantages	**SNMPv3 advantages**
SNMPv1 and SNMPv2c allow particular IP addresses to access all data or no data	SNMPv3 allows a particular user to access particular data
Not very robust (password sent in PDU)	Robust (data integrity and data origin authentication)
Any user that can read data can also change the data (for objects defined as read-write).	The ability to change data can be limited to specific users
No data confidentiality	Encryption available (separate product)
Configuration changes require restarting of SNMP agent	Configuration changes for USM and VACM can be made dynamically, either locally or remotely

For more information about security, see “Creating User Keys” on page 847.

Complete the following steps to configure SNMP:

1. Configure the SNMP agent (OSNMPD).
2. Configure the SNMP commands:
   - SNMP for NetView SNMP
   - osnmp for OS/390 UNIX SNMP
3. Configure the SNMP subagents.
4. Configure the ATM Open Systems Adapter 2 (ATM OSA-2) support.

Step 1: Configure the SNMP Agent (OSNMPD)

Configure the SNMP agent based upon your security need. The SNMP agent accepts both SNMPv1 and SNMPv2c requests for community-based security. The SNMP agent can be configured to also use the User-based Security Model and the View-based Access Control Model. To configure the SNMP agent, perform the following tasks:

- “Provide TCP/IP Profile Statements” on page 826

- Depending upon whether you want to use USM and VACM, do one of the following:
  - If you are using community-based security and do not need USM or VACM, see “Provide Community-Based Security and Trap Destination Information” on page 827.
  - If you want the flexibility of using USM or VACM or community-based security, see “Provide Community-Based and User-Based Security and Trap Destination Information” on page 830.
- “Provide MIB Object Configuration Information” on page 850
- “Start the SNMP Agent (OSNMPD)” on page 852

Provide TCP/IP Profile Statements

Update the following configuration statements in *hlq*.PROFILE.TCPIP:
AUTOLOG
PORT

There are two primary TCP/IP ports used by the SNMP agent, one for receiving incoming requests and one for sending traps to managers.

The default port used by the SNMP agent to receive incoming requests is 161. If you want the agent to use port 161 for this purpose and want to insure that no other application uses this port, you must specify the following PORT statement in your profile data set:

```
PORT
  161 UDP OSNMPD              ; SNMP Agent port for SNMP requests
```

If the agent will be started from the OS/390 shell, reserve the port instead for OS/390 UNIX by typing *OMVS* instead of *OSNMPD*.

If you want to define a port other than 161 for SNMP requests, you must do the following:

1. Start the agent with a `-p` parameter.
2. Configure management applications to use the new port:
   - Make an entry in the OSNMP.CONF file with the correct port number. For details on creating this entry, see the description for *targetAgent* in “OSNMP.CONF Statement Syntax” on page 862.
   - Where supported, configure other management applications to use the new port. The NetView SNMP command does not allow use of any port other than the default port.
3. Configure subagents to use the new port:
   a. Specify the port number to use on the SACONFIG profile statement for the TCP/IP subagent
   b. Specify the port number to use on the ROUTESA_CONFIG profile statement for the OMPROUTE subagent
   c. Specify the port number to use on the -p parameter when starting the SLA subagent
   d. If you are using DPI subagents other than those supplied with CS for OS/390, set the SNMP_PORT environment variable to enable user-written subagents to connect to the agent.

The SNMP agent uses port 162, by default, for sending traps to the managers specified in SNMPTRAP.DEST or SNMPD.CONF file. If you plan to listen for traps from both NetView SNMP and OS/390 UNIX SNMP at the same IP address, you must specify the following PORT statement in your profile data set:

```
PORT
  162 UDP SNMPQE  ; SNMPQuery Engine
```

You must also reserve additional ports for use by the osnmp command by specifying

```
nnnnn UDP OMVS
```

where *nnnnn* is a number in the range 0–65535 and *nnnnn* is used as the -p parameter value on the osnmp trap command.

If you want the SNMPQE and OSNMPD address spaces to be started automatically when the TCPIP address space is started, then include SNMPQE and OSNMPD in the AUTOLOG statement:

```
AUTOLOG
  SNMPQE                    ; SNMP Query Engine
  OSNMPD                    ; SNMP Agent
ENDAUTOLOG
```

Provide Community-Based Security and Trap Destination Information

If you are using only community-based security without the view-based access control model, do the following to configure the security and trap destinations.

Provide Community Name Information

SNMP agents are accessed by remote network management stations. To allow network management stations to send inquiries to the SNMP agent, you may provide PW.SRC information that defines a list of community names and IP addresses that can use these community names. The community name operates as a password when accessing objects on a destination SNMP agent.

The PW.SRC information is optional. If no PW.SRC information is foundand no community name is specified for the -c parameter at agent invocation, then the SNMP agent will accept requests with a community name of 'public' from any IP address. If a PW.SRC file exists, but is empty, and if no community name is specified on the `-c` parameter at the agent invocation, then no requests will be accepted by the agent.

PW.SRC Search Order: To access the PW.SRC information, the search order is:

1. /etc/pw.src HFS file
2. The data set specified on SYSPWSRC DD statement in the agent procedure
3. *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent
4. SYS1.TCPPARMS(PWSRC)
5. *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used.

The first file found in the search order is used.

Note: Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the PW.SRC file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

PW.SRC Statement Syntax:

- The PW.SRC statements specify community names and hosts that can use each community name. The format of a statement is:

  ```
  community_name  desired_network  snmp_mask
  ```

  The *community_name* can be up to 32 characters in length. This value can contain both uppercase and lowercase 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 the correct case.
- All parameters for each community must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with either an asterisk (*) or a pound sign (#).

PW.SRC Example: The PW.SRC statements could be specified as follows:

```
passwd1 9.0.0.0      255.0.0.0
passwd2 129.34.81.22 255.255.255.255
```

The community name of an incoming SNMP request is compared to the known community names. If a match is found, then the IP address of the incoming request is logically ANDed with the *snmp_mask* of the PW.SRC statement. The result of the logical ANDing process is compared with the *desired_network*. If they match, the request is accepted.

In the preceding example, if a request for *community_name* `passwd1` is received from the IP address 9.34.22.122, IP address 9.34.22.122 is ANDed with 255.0.0.0. The result is 9.0.0.0, which equals the specified *desired_network* for `passwd1`, so this request is accepted. In `passwd2`, if the *community_names* match, only requests from host 129.34.81.22 are accepted.

If the *community_name* values do not match, or the IP address ANDed with the *snmp_mask* does not match, an `AUTHENTICATION_FAILURE` trap is sent if both of the following are true:

- A destination entry exists in SNMPTRAP.DEST
- Authentication failure traps have been enabled. These traps are enabled by setting MIB object ″snmpEnableAuthenTraps.0″ to 1.

A *desired_network* and *snmp_mask* of all zeros allows anyone with the correct *community_name* to make requests.

Provide Trap Destination Information

Traps are unsolicited messages that are sent by an SNMP agent 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 agent.

The following traps are agent-generated up through PVC deletion:

- AUTHENTICATION_FAILURE
- COLD_START
- LINK_DOWN
- LINK_UP

- PVC creation
- PVC deletion

  PVC traps are reported for ATM Permanent Virtual Connections. For further information, see “Step 4: Configure the ATM Open Systems Adapter 2 (ATM OSA-2) Support” on page 867.

The following traps are generated by the SLA subagent:

- Monitored event not achieved for aggregate policy
- Monitored event okay for aggregate policy
- Monitored even not achieved for individual connection policy
- Monitored event okay for individual connection policy
- Policy deleted
- Monitor table entry deleted

Note: The SNMP agent Distributed Protocol Interface allows subagents other than those shipped with CS for OS/390 (which might be running on another host) to generate SNMP traps. This can allow for support of other types of traps. For more information about SNMP DPI, see the *OS/390 SecureWay Communications Server: IP Programmer's Reference*.

To use traps, you must provide SNMPTRAP.DEST information defining a list of managers to which traps are sent. The SNMPTRAP.DEST information is optional. If no trap destination file is found, then the SNMP agent sends traps to the IP address of the SNMP agent and issues a warning message indicating that defaults are in effect. If a trap destination file exists, but is empty, no traps are sent.

SNMPTRAP.DEST Search Order: To access the SNMPTRAP.DEST information, the search order is:

1. /etc/snmptrap.dest HFS file
2. The data set specified on SNMPTRAP DD statement in the agent procedure
3. *jobname*.SNMPTRAP.DEST, where *jobname* is the name of the job used to start the SNMP agent
4. SYS1.TCPPARMS(SNMPTRAP)
5. *hlq*.SNMPTRAP.DEST, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used.

The first file found in the search order will be used.

Note: Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

SNMPTRAP.DEST Statement Syntax:

- The SNMPTRAP.DEST statements list managers who are to receive the traps, and the protocol used to send traps. The format of a statement is:

  ```
  manager UDP
  ```

  The *manager* is the host to which the trap is to be sent. This can be a host name, or it can be the IP address of the host. If a host name is specified, the value may contain both uppercase and lowercase letters and it is not

case-sensitive. The protocol must be UDP. There should be one entry in the data set for each host to which you want to send traps.

- All parameters for each host must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with an asterisk (*) or a pound sign (#).

SNMPTRAP.DEST Example: The SNMPTRAP.DEST statements could be specified as follows:

```
# SNMP Trap Destination information
124.34.216.1 UDP
MVSSYS2      UDP
```

Provide Community-Based and User-Based Security and Trap Destination Information

If you want to use user-based security, either concurrently with or instead of community-based security, you must configure security definitions and trap destinations.

Note: If not already familiar with configuring security definitions and trap destinations in previous releases, first read “Provide Community-Based Security and Trap Destination Information” on page 827.

SNMPv3 provides the ability to configure the agent dynamically, from either a local or remote host, and to make changes in the configuration while the SNMP agent is running. Doing SNMP agent configuration dynamically requires a good understanding of how the SNMP SET commands can be issued to create new rows or to change or delete existing rows, as well as familiarity with the SNMP engine configuration tables defined in RFCs 2271 through 2275. (For additional detail, see RFCs 2271 through 2275, as well as RFCs 1901 through 1910.)

As an alternative to dynamically configuring the SNMP agent, CS for OS/390 supports a configuration file to be read at agent initialization called the SNMPD.CONF file. Dynamic configuration changes made with SNMP SET commands to the SNMP agent configuration entries will be written out to the SNMPD.CONF file, so they will continue to be in effect even after the SNMP agent is restarted.

SNMPD.CONF File

The SNMPD.CONF file defines the SNMP agent security and notification destinations. (A trap is one type of notification. The other type, informs, are not supported in CS for OS/390.) If the SNMPD.CONF file exists, the agent can support SNMPv1, SNMPv2c, and SNMPv3 requests. If no SNMPD.CONF file exists, the agent will support only SNMPv1 and SNMPv2c requests.

SNMPD.CONF Search Order: To access SNMPD.CONF information, the search order is:

1. The name of an HFS file or an MVS file specified by the SNMPD_CONF environment variable.
2. /etc/snmpd.conf

Note: If the SNMPD.CONF file is not provided, then no USM or VACM configuration is done and the files, PW.SRC and SNMPTRAP.DEST, will be

sought. If the SNMPD.CONF file is found, the PW.SRC file and SNMPTRAP.DEST files will not be used.

SNMPD.CONF Dynamic Configuration: If the SNMPD.CONF information is to be located in an MVS data set rather than an HFS file, special considerations must be made to support dynamic configuration changes to the SNMP agent's configuration. If dynamic configuration changes are made, the file is rewritten to reflect the changes. Therefore, consider the following when allocating the SNMPD.CONF file to an MVS data set:

- The record length (LRECL) should be 512 bytes to accomodate the longest possible entry.
- The use of a member of a partitioned data set is tolerated but not recommended. Because the file might be rewritten often, frequent compression of the partitioned data set may become necessary. In addition, locking on the file is done at the data set level, not at the member level, so other members of the partitioned data set would not be usable while the SNMP agent was running (once a dynamic configuration change had been made).

SNMPD.CONF Statement Syntax: The SNMPD.CONF file supports nine different types of entries:

USM_USER
Defines a user for the User-Based Security Model (USM).

VACM_GROUP
Defines a security group (made up of users or communities) for the View-Based Access Control Model (VACM).

VACM_VIEW
Defines a particular set of MIB objects, called a view, for the View-Based Access Control Model.

VACM_ACCESS
Identifies the access permitted to different security groups for the View-Based Access Control Model.

NOTIFY
Identifies management targets to receive notifications.

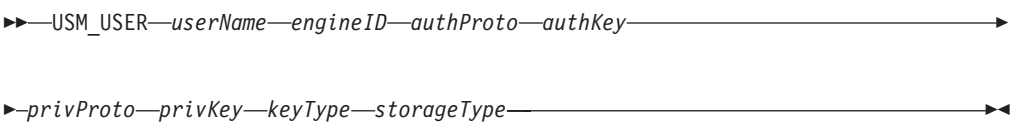
TARGET_ADDRESS
Defines a management application's address and identifies parameters to be used in sending notifications.

TARGET_PARAMETERS
Defines the message processing and identifies security parameters to be used in sending notifications to a particular management target.

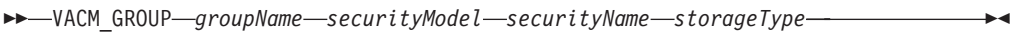
COMMUNITY
Defines a community for community-based security.

DEFAULT_SECURITY
Identifies the default security posture to be configured for the SNMP agent; additional security definitions defined by the use of the preceding eight entry definition types augment any default security configurations defined as a result of the DEFAULT_SECURITY statement.

Defining the USM_USER Entry

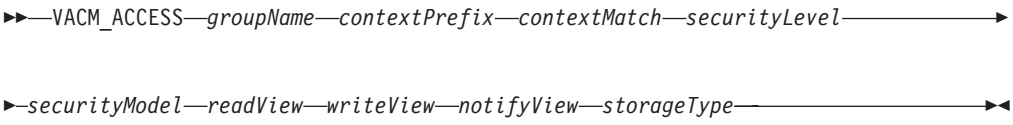


Defining the VACM_GROUP Entry



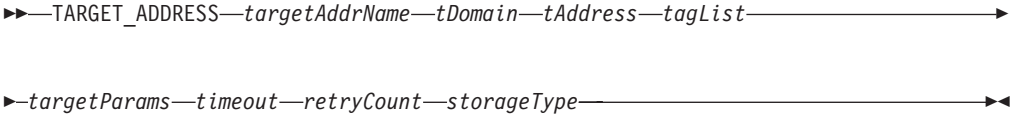
Defining the VACM_VIEW Entry

Defining the VACM_ACCESS Entry

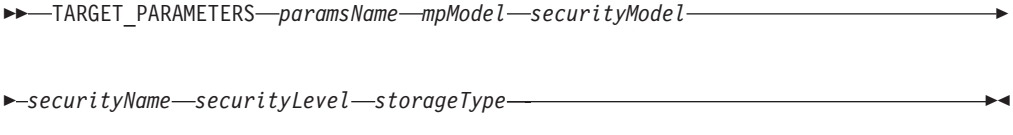


Defining the NOTIFY Entry

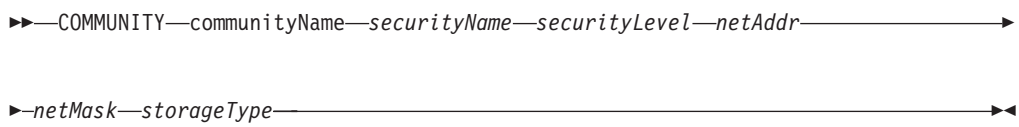
Defining the TARGET_ADDRESS Entry



Defining the TARGET_PARAMETERS Entry



Defining the COMMUNITY Entry



Defining the DEFAULT_SECURITY Entry

Parameters:

USM_USER Entry:

USM_USER userName engineID authProto authKey privProto privKey keyType storageType

Field definitions:

- *userName* indicates the name of the user for the User-Based Security Model (USM). The userName must be unique to the SNMP agent. The userName is used as the security name for the User-based Security Model; the contents of this field will be used as the securityName value for other entries (such as the VACM_GROUP entry) when the securityModel is USM. The userName field is a 1–32 character string. There is no default value.
- *engineID* indicates the engineID of the authoritative side of the message. The engineID for the CS for OS/390 SNMP agent is determined at agent initialization; it is either read in from the SNMPD.BOOTS file or it is generated automatically and stored in the SNMPD.BOOTS file. It can be retrieved dynamically by issuing a get request for object snmpEngineID.0. For get, getbulk, set, response, and trap messages, the authoritative side is the SNMP agent. For inform messages, the authoritative side is the notification receiver. Note, however, that CS for OS/390 does not support informs. Valid values are a string of 2–64 hexadecimal digits or a '-' (dash) to indicate the default value (the local SNMP agent's engineID).
- *authProto* indicates the authentication protocol to be used on authenticated messages on behalf of this user. Valid values are 'HMAC-MD5', 'HMAC-SHA', 'none' to indicate that no authentication is to be done, or a '-' (dash) to indicate the default value HMAC-MD5.
- *authKey* indicates the authentication key to be used in authenticating messages on behalf of this user. This field is ignored when authProto is specified as 'none'. The keyType field will indicate whether the key is localized or non-localized. Valid values are 32 hexadecimal digits when authProto is 'HMAC-MD5' and 40 hexadecimal digits when authProto is 'HMAC-SHA'. A '-' (dash) indicates the default (no key, indicating no authentication). For information on generating authentication and privacy keys using the pwtokey command, see "Creating User Keys" on page 847.
- *privProto* indicates the privacy protocol to be used on encrypted messages on behalf of this user. Privacy can be requested only if authentication is also requested. If authentication is not requested, this field is ignored. Valid values are 'DES' to indicate CBC 56-bit DES (only with the additional encryption product), 'none' to indicate no privacy, and a '-' (dash) to indicate the default of no privacy.
- *privKey* is used in authenticating messages to and from this user. This field is ignored when privProto is specified or defaulted as 'none'. The keyType field indicates whether the key is localized or non-localized. Privacy can be requested

only if authentication is also requested. If authentication is not requested, this field is ignored. The privacy key and the authentication key are assumed to have been generated using the same authentication protocol (HMAC-MD5 or HMAC-SHA). Valid values are 32 hexadecimal digits if the key is localized or if the key is non-localized and the authProto is 'HMAC-MD5', 40 hexadecimal digits if the key is non-localized and the authProto is 'HMAC-SHA', or a '-' (dash) to indicate the default of no key, indicating no encryption. For information on generating privacy keys using the pwtokey command, see "Creating User Keys" on page 847.

- *keyType* indicates whether the keys defined by authKey and privKey are localized or non-localized. Localized indicates that they have been generated with the appropriate engineID, making the key usable only at one snmpEngine. Non-localized indicates the key can be used at different snmpEngines. The authKey and privKey, if both are specified, must both be localized or both be non-localized. This field is ignored if no authentication or privacy is requested. Valid values are 'L' to indicate keys are localized, 'N' to indicate keys are non-localized, or a '-' (dash) to indicate the default value of localized.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.
  - 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests. For the USM_USER entry, readOnly is not allowed unless the authentication protocol is 'none' because protocols require that user keys be changeable.
  - '-' (dash)

    Indicates the default value of non-volatile.

VACM_GROUP Entry:

```
VACM_GROUP groupName securityModel securityName storageType
```

Field definitions:

- *groupName* is the group name for the View-Based Access Control Model (VACM). The groupName must be specified; there is no default value. It must be a 1–32 character string.
- *securityModel* indicates the SNMP security model for this entry. When an SNMP message comes in, the securityModel and the securityName are used to determine the group to which the user (or community) represented by the securityName belongs. Valid values are 'SNMPv1' to indicate community-based security using SNMPv1 message processing, 'SNMPv2c' to indicate community-based security using SNMPv2c message processing, 'USM' to indicate the User-Based Security Model, or a '-' (dash) to indicate the default value of USM.
- *securityName* indicates a member of this group. Valid values are 1–32 characters indicating a USM userName when securityModel is 'USM', or a community Name when securityModel is 'SNMPv1' or 'SNMPv2c'. There is no default value.

- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.
  - 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.
  - '-' (dash)

    Indicates the default value of non-volatile.

VACM_VIEW Entry:

```
VACM_VIEW viewName viewSubtree viewMask viewType storageType
```

Field definitions:

- *viewName* indicates the textual name of the view for the View-Based Access Control Model. View names do not need to be unique. Multiple entries with the same name together define one view. However, the viewname, together with the subtree object ID, must be unique to an SNMP engine. Valid values are 1–32 characters in length. There is no default value.
- *viewSubtree* indicates the MIB object prefix of the MIB objects in the view. Valid values are an object ID of up to 128 sub-OIDs, a textual object name (or object prefix), or a combination of textual object name followed by numeric sub-OIDs. The name must be found within the compiled MIB or in the logical extension to the MIB, the /etc/mibs.data file. There is no default value.
- *viewMask* indicates a mask that specifies which of the sub-OIDs in the subtree are relevant. See RFC2275 for further information on the viewMask. Valid values are a hex string of up to 16 bytes (up to 128 bits), where each hexadecimal digit represents four bits. Each bit indicates whether or not the corresponding sub-OID in the subtree is relevant, or a '-' (dash) to indicate the default value (a mask of all ones meaning all sub-OIDs are relevant).
- *viewType* indicates the type of the view definition. Valid values are 'included' to indicate the MIB objects identified by this view definition are within the view, 'excluded' to indicate the MIB objects identified by this view definition are excluded from the view, or a '-' (dash) to indicate the default value of 'included'.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.
  - 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

– '-' (dash)

Indicates the default value of non-volatile.

VACM_ACCESS Entry:

```
VACM_ACCESS groupName contextPrefix contextMatch securityLevel
securityModel readView writeView notifyView storageType
```

Field definitions:

- *groupName* is the group name for the View-Based Access Control Model (VACM) for which access is being defined. The groupName must be specified; there is no default value. It must be a 1–32 character string.
- *contextPrefix* indicates a character string to be compared with the incoming contextName, if the value specified for the contextMatch field is 'prefix'. Note, however, that the SNMP agent in CS for OS/390 supports MIB objects in only the local (null) context. Valid values are 1–32 characters, or a dast (-) to indicate the default value of the null context (″″).
- *contextMatch* indicates whether the incoming contextName must be compared with (and match exactly) the entire contextName or whether only the first part of the contextName (up to the length of the indicated value of the contextPrefix) must match. Valid values are 'exact' to indicate that the entire contextName must match, 'prefix' to indicate that only the prefix of the contextName must match, or a '-' (dash) to indicate the default value of 'exact'.
- *securityLevel* indicates the securityLevel for this entry and is used in determining which access table entry to use. Valid values are 'noAuthNoPriv' or 'none' to indicate no authentication or privacy protocols are applied, 'AuthNoPriv' or 'auth' to indicate authentication protocols are applied but no privacy protocol is applied, 'AuthPriv' or 'priv' to indicate both authentication and privacy protocols are applied (If the additional encryption product is not applied, this level can be configured but cannot actually be used), or a '-' (dash) to indicate the default value of 'noAuthNoPriv'.
- *securityModel* indicates the SNMP security model for this entry and is used in determining which access table entry to use. Valid values are 'SNMPv1' to indicate community-based security using SNMPv1 message processing, 'SNMPv2c' to indicate community-based security using SNMPv2c message processing, 'USM' to indicate the User-Based Security Model, or a dash '-' (dash)to indicate the default value of USM.
- *readView* indicates the name of the view to be applied when read operations (get, getnext, getbulk) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a '-' (dash) to indicate the default value of no view (no readView defined for members of this group).
- *writeView* indicates the name of the view to be applied when write operations (set) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a '-' (dash) to indicate the default value of no view (no writeView defined for members of this group).
- *notifyView* indicates the name of the view to be applied when notify operations (traps or informs) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a '-' (dash) to indicate the default value of no view (no notifyView defined for members of this group).
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:

– 'nonVolatile'

  Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

– 'permanent'

  Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

– 'readonly'

  Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

– '-' (dash)

  Indicates the default value of non-volatile.

NOTIFY Entry:

```
NOTIFY notifyName tag type storageType
```

Field definitions:

- *notifyName* is a locally unique identifier for this notify definition. Valid values are 1–32 characters. There is no default value.
- *tag* indicates a tag value to be compared with the values in the tagLists defined in the snmpTargetAddrTable (either on TARGET_ADDRESS entries or via dynamic configuration). For each match of this tag with a value in the tagLists defined in the snmpTargetAddrTable, a notification may be sent. See RFC2273 for a definition of SnmpTagValue. Valid values are 1–255 characters. No delimiters are allowed. A '-' (dash) indicates the default, which is no tag value.
- *type* indicates which type of notification should be generated. Valid values are 'trap' to indicate a trap (an unconfirmed notification; notification sent with trap PDUs) or a '-' (dash) to indicate the default value of 'trap'. Inform type notifications are not supported in CS for OS/390.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  – 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

  – 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  – 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  – '-' (dash)

    Indicates the default value of non-volatile.

TARGET_ADDRESS Entry:

```
TARGET_ADDRESS targetAddrName tDomain tAddress tagList targetParams
timeout retryCount storageType
```

Field definitions:

- *targetAddrName* indicates a locally unique identifier for this target address definition. Valid values are 1–32 characters in length. There is no default value.

- *tDomain* indicates the transport type of the address indicated by tAddress. Valid values are 'UDP' for UDP datagrams or a '-' (dash) for the default value of 'UDP'.
- *tAddress* indicates the transport address to which notifications are sent. Valid values are a 1–21 character octet string indicating the IP address and optionally the UDP port in the form ip_address:port (for example, 9.37.84.48:162). The IP address must be specified as a.b.c.d where a, b, c, and d are in the range of 0–255, and the port (if specified) must be in the range 1–65535. The IP address may not be defaulted, but the port, if not specified, will default to 162.
- *tagList* indicates a list of tag values that are used to select target addresses for a notification operation. The CS for OS/390 implementation supports, via the configuration file, only one tag in a tagList. RFC2273 contains the complete definition of SnmpTagList and SnmpTagValue. The CS for OS/390 implementation accepts as valid values a string of 1–255 characters. No delimiters are allowed. A '-' (dash) indicates the default value, an empty list.
- *targetParams* indicates a TARGET_PARAMETERS paramsName value that indicates which security and message processing is to be used in sending notifications to this target. Valid values are 1–32 characters in length. There is no default value.
- *timeout* indicates the expected maximum round trip time for communicating with this target address (in 1/100ths of a second). Valid values are 0–2147483647, or a '-' (dash) to indicate the default value of 0 (meaning no timeout value). Timeout is used only for inform type notifications; it is not used for traps. Since only traps are supported on CS for OS/390, this value is ignored.
- *retryCount* indicates the number of retries to be attempted when a response is not received for a generated message. Valid values are 0–255, or a '-' (dash) to indicate the default value of 0 (meaning no retry). RetryCount is used only for inform type notifications; it is not used for traps. Since only traps are supported on CS for OS/390, this value is ignored.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.
  - 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.
  - '-' (dash)

    Indicates the default value of non-volatile.

TARGET_PARAMETERS Entry:

```
TARGET_PARAMETERS paramsName mpModel securityModel securityName
securityLevel storageType
```

Field definitions:

- *paramsName* is a locally unique identifier for this target parameters definition. Valid values are 1–32 characters in length. There is no default value.

- *mpModel* is the message processing model to be used in sending notifications to targets with this parameter definition. Valid values are 'SNMPv1', 'SNMPv2c', and 'SNMPv3'. There is no default value.
- *securityModel* indicates the security model to be used in sending notifications to targets with this parameter definition. Valid values are 'SNMPv1', 'SNMPv2c', or 'USM' to indicate User-Based Security Model. There is no default value.
- *securityName* identifies the principal (user or community) on whose behalf SNMP messages will be generated using this parameter definition. For community-based security, this is a community name. For USM, this is a user name. Valid values are 1–32 characters in length. There is no default value.
- *securityLevel* indicates the security level to be used in sending notifications to targets with this parameter definition. Valid values are 'noAuthNoPriv' or 'none' to indicate no authentication or privacy protocols are applied, 'AuthNoPriv' or 'auth' to indicate that authentication protocols are applied but no privacy protocol is applied, 'AuthPriv' or 'priv' to indicate that both authentication and privacy protocols are applied (If the additional encryption product is not applied, this level can be configured, but not actually used), or a '-' (dash) to indicate the default value of 'noAuthNoPriv'.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - 'nonVolatile'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.
  - 'permanent'

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - 'readonly'

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.
  - '-' (dash)

    Indicates the default value of non-volatile.

COMMUNITY Entry:

```
COMMUNITY communityName securityName securityLevel netAddr netMask storageType
```

Field definitions:

- *communityName* is the community name for community-based security (SNMPv1 or SNMPv2c). The communityName must be specified; there is no default value. It must be a 1–32 character string.
- *securityName* is the securityName defined for this communityName. The securityName is the more generic term for the principal (user or community) for which other entries, such as VACM_GROUP and TARGET_PARAMETERS, are defined. The community name must match the securityName exactly. The securityName is 1–32 characters. A '-' (dash) indicates the default value; a securityName equal to the specified communityName.
- *securityLevel* indicates the security level to be applied when processing incoming or outgoing messages with this community name. Valid values are 'noAuthNoPriv' or 'none' to indicate no authentication or privacy protocols are applied, or '-' (dash) to indicate the default value of 'noAuthNoPriv'. Encryption is not supported on SNMPv1 and SNMPv2c messages.

- *netAddr* is the IP address in dotted decimal notation representing the range of addresses for which this communityName may be used. The netAddr must be specified; there is no default value.

  Valid values are in the form a.b.c.d, where a, b, c, and d are in the range 0–255.
- *netMask* is the IP address mask to be logically ANDed with the origin address of the incoming SNMP message. If the resulting value equals the value of netAddr, the incoming message is accepted. netMask must be specified; there is no default value.

  Valid values are in the form a.b.c.d, where a, b, c, and d are in the range 0–255.
- *storageType* indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC1903. Note that the value of 'volatile' is not supported in the SNMPD.CONF file. Valid values are:
  - nonVolatile

    Indicates the entry definition will persist across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.
  - permanent

    Indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.
  - readonly

    Indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.
  - '-' (dash)

    Indicates the default value of non-volatile.

DEFAULT_SECURITY Entry:

```
DEFAULT_SECURITY securityPosture password privacy
```

Field definitions:

- *securityPosture* indicates the default security posture to be configured for the SNMP agent, as defined by Appendix A of RFC 2275 (and outlined below). Valid values are 'minimum-secure' to indicate the SNMP agent will be configured with the least secure default configurations, 'semi-secure' to indicate the SNMP agent will be configured with moderately secure default configurations, and 'no-access' to indicate the SNMP agent will be configured with no default configurations. The default value is 'no-access'.

  Following are the default security definitions based on the selected security posture:
  - no-access

    No initial configurations are done.
  - semi-secure

    If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified:

    ```
    USM_USER initial- HMAC-MD5 ### none - N permanent
    ```

    where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

    If privacy is requested (and available with the additional encryption product), a default user is configured as if the following USM_USER entry had been specified:

    ```
    USM_USER initial - HMAC-MD5 ### DES ### N permanent
    ```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified:

```
VACM_GROUP initial USM initial readOnly
```

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

```
VACM_ACCESS initial - exact none   USM restricted -         restricted readOnly
VACM_ACCESS initial - exact auth   USM internet   internet  internet   readOnly
VACM_ACCESS initial - exact priv   USM internet   internet  internet   readOnly
```

Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

```
VACM_VIEW internet   internet     - included readOnly
VACM_VIEW restricted system       - included readOnly
VACM_VIEW restricted snmp         - included readOnly
VACM_VIEW restricted snmpEngine   - included readOnly
VACM_VIEW restricted snmpMPDStats - included readOnly
VACM_VIEW restricted usmStats     - included readOnly
```

– minimum-secure

If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial - HMAC-MD5 ### none - N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

If privacy is requested (and available with the additional encryption product), a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial - HMAC-MD5 ### DES ### N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified:

```
VACM_GROUP initial USM initial readOnly
```

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

```
VACM_ACCESS initial - exact none   USM restricted -         restricted readOnly
VACM_ACCESS initial - exact auth   USM internet   internet  internet   readOnly
VACM_ACCESS initial - exact priv   USM internet   internet  internet   readOnly
```

Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

```
VACM_VIEW internet   internet     - included readOnly
VACM_VIEW restricted internet     - included readOnly
```

- *password* indicates the password to be used to generate authentication and privacy keys for user 'initial'. If 'no-access' is specified as the securityPosture, this keyword is ignored. Valid values are a string of 8–255 characters, or a '-' (dash) to indicate the default value (no password). The default is only accepted if securityPosture is 'no-access'.

- *privacy* indicates whether or not encryption is to be supported for messages on behalf of user 'initial'. Valid values are 'Yes' to indicate that privacy is supported for user 'initial' (only with the additional encryption product), 'No' to indicate that privacy is not supported for user 'initial', or a '-' (dash) to indicate the default value of 'no'. If 'no-access' is selected as the security posture, this value will be ignored.

Usage Notes:

- An entry must be contained on one line.
- Keywords in the SNMPD.CONF file are accepted in any case (that is, they are *not* case sensitive.)
- Values in the SNMPD.CONF file *are* case-sensitive. (for example, a userName of user1 is not equivalent to a userName of USER1.)
- All of the entry definitions above require that *all* fields be specified, either with a specific value or with a dash ('-'). A dash indicates that the appropriate default value should be applied.
- If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is generated.
- Statements in the SNMPD.CONF file are not order dependent. However, if more than one DEFAULT_SECURITY statement is found, the last one in the file is the one that is used.
- If there are no valid entries in the SNMPD.CONF file (but the file exists) and no -c parameter specified at agent invocation, no requests are accepted.
- Comments may be entered in the SNMPD.CONF file. They must must begin with an asterisk (*) or a pound sign (#) in column 1.

SNMPD.CONF Example: A sample SNMPD.CONF file is shipped as /usr/lpp/tcpip/samples/snmpd.conf. Following are sample entries for the SNMPD.CONF file.

```
# snmpd.conf sample
#
# Sample file showing format of configuration file for the SNMP agent
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5647-A01
# (C) Copyright IBM Corp. 1997, 1998
# Status = CSV2R7
#
#-----------------------------------------------------------------------------------------------------------------------------------------------
# Notes
# - All values for an entry must be on the same line.
# - All keys need to be regenerated using the pwtokey command in order
#   for these sample entries to actually be used.
# - In this sample:
#   - Keys are generated for use with engineID 00000002000000000943714F
#   - Authentication keys were generated with password of
#     username+"password", such as "u1password"
#   - Privacy keys were generated with password of
#     username+"privpass", such as "u1privpass"
#   - Entries defined to use encryption support, which is available only
#     as a separately orderable feature on the base OS/390 product, are
#     included below but commented out.
#-----------------------------------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------------------------------------------------------
# USM_USER entries
# Format is:
#  userName engineID authProto authKey privProto privKey keyType storageType
#
# Note: Users u3 and u4 use non-localized keys. Not recommended, but allowed.
# Note: Users u5 and u6 use the same password for generating the authkey and privkey.  Not recommended, but allowed.
#-----------------------------------------------------------------------------------------------------------------------------------------------
*USM_USER u1 - HMAC-MD5 6da6c69c64b7737360f8319d90e4d511         DES 959709e534eade82cecbacb42a10c90a         L -
*USM_USER u2 - HMAC-SHA f26562da268f21a916792d3f45500cd5a8163071 DES 3b3249ad3eb10d46d2731ee6fbaf8591         L -
*USM_USER u3 - HMAC-MD5 d1f86e9c9346253c10a4cd2da339b1db         DES cfccde3249ba521ae4da0ddfc2b76ee7         N -
*USM_USER u4 - HMAC-SHA 42529b3c6c138c173e70db1050de8d74c04205cb DES ef70a0a98d399a9189f3169e82010f3b46e694e2 N -
*USM_USER u5 - HMAC-MD5 2b0e2c55d452b5ada056d50e8a66ea35         DES 2b0e2c55d452b5ada056d50e8a66ea35         L -
*USM_USER u6 - HMAC-SHA aaa2f3b36f840549b6e8916b7b90430765dd3858 DES aaa2f3b36f840549b6e8916b7b904307         L -
USM_USER  u1 - HMAC-MD5 6da6c69c64b7737360f8319d90e4d511         -   -                                        L -
USM_USER  u2 - HMAC-SHA f26562da268f21a916792d3f45500cd5a8163071 -   -                                        L -
USM_USER  u3 - HMAC-MD5 d1f86e9c9346253c10a4cd2da339b1db         -   -                                        N -
USM_USER  u4 - HMAC-SHA 42529b3c6c138c173e70db1050de8d74c04205cb -   -                                        N -
USM_USER  u5 - HMAC-MD5 2b0e2c55d452b5ada056d50e8a66ea35         -   -                                        L -
USM_USER  u6 - HMAC-SHA aaa2f3b36f840549b6e8916b7b90430765dd3858 -   -                                        L -

#-----------------------------------------------------------------------------------------------------------------------------------------------
# VACM_GROUP entries
# Format is:
#  groupName securityModel securityName storageType
#-----------------------------------------------------------------------------------------------------------------------------------------------
VACM_GROUP group1 USM     u1 -
VACM_GROUP group1 USM     u2 -
VACM_GROUP group1 USM     u3 -
VACM_GROUP group1 USM     u4 -

VACM_GROUP group2 USM     u5 -
VACM_GROUP group2 USM     u6 -

VACM_GROUP group3 SNMPv1  publicv1  -
VACM_GROUP group3 SNMPv2c publicv2c -

VACM_GROUP group4 SNMPv1  MVSsubagent  -
VACM_GROUP group4 SNMPv2c MVSsubagent  -
```

Figure 37. SNMPD.CONF File - Part 1 of 3

```
#-------------------------------------------------------------------------------------------------------------------------------------------------------
# VACM_VIEW entries
# Format is:
#  viewName viewSubtree viewMask viewType storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
VACM_VIEW bigView        internet   - included -

VACM_VIEW prettyBigView  internet   - included -
VACM_VIEW prettyBigView  interfaces - excluded -

VACM_VIEW mediumView     system     - included -
VACM_VIEW mediumView     interfaces - included -
VACM_VIEW mediumView     tcp        - included -
VACM_VIEW mediumView     udp        - included -
VACM_VIEW mediumView     icmp       - included -

VACM_VIEW smallView      snmp       - included -

VACM_VIEW subagentView   dpiPort    - included -

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# VACM_ACCESS entries
# Format is:
#  groupName contextPrefix contextMatch securityLevel securityModel readView writeView notifyView storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
*VACM_ACCESS group1 - - AuthPriv     USM     bigView        bigView       bigView   -
VACM_ACCESS  group1 - - AuthNoPriv   USM     bigView        prettyBigView smallView -
VACM_ACCESS  group1 - - noAuthNoPriv USM     smallView      smallView     smallView -

*VACM_ACCESS group2 - - AuthPriv     USM     bigView        bigView       bigView   -
VACM_ACCESS  group2 - - AuthNoPriv   USM     bigView        mediumView    smallView -
VACM_ACCESS  group2 - - noAuthNoPriv USM     bigView        mediumView    -         -

VACM_ACCESS  group3 - - noAuthNoPriv SNMPv1  mediumView     mediumView    mediumView -
VACM_ACCESS  group3 - - noAuthNoPriv SNMPv2c bigView        bigView       bigView   -

VACM_ACCESS  group4 - - noAuthNoPriv SNMPv1  subagentView   -             -         -
VACM_ACCESS  group4 - - noAuthNoPriv SNMPv2c subagentView   -             -         -

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# NOTIFY entries
# Format is:
#  notifyName tag type storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
NOTIFY notify1 traptag trap -

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# TARGET_ADDRESS
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout retryCount storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
TARGET_ADDRESS Target1 UDP 9.67.113.10     traptag trapparms1 - - -
TARGET_ADDRESS Target2 UDP 9.67.113.5:2162 traptag trapparms2 - - -
TARGET_ADDRESS Target3 UDP 127.0.0.1       traptag trapparms3 - - -
```

Figure 38. SNMPD.CONF File - Part 2 of 3

```
#-------------------------------------------------------------------------------------------------------------------------------------------------------
# TARGET_PARAMETERS
# Format is:
#  paramsName mpModel securityModel securityName securityLevel storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  publicv1  noAuthNoPriv -
TARGET_PARAMETERS trapparms2 SNMPv2c SNMPv2c publicv2c noAuthNoPriv -
*TARGET_PARAMETERS trapparms3 SNMPv3  USM     u3        AuthPriv -

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# COMMUNITY
# Format is:
#  communityName securityName securityLevel netAddr netMask storageType
#-------------------------------------------------------------------------------------------------------------------------------------------------------
COMMUNITY publicv1    publicv1     noAuthNoPriv 9.67.113.79 255.255.255.255 -
COMMUNITY publicv2c   publicv2c    noAuthNoPriv 0.0.0.0     0.0.0.0         -
COMMUNITY MVSsubagent MVSsubagent  noAuthNoPriv 9.0.0.0     255.0.0.0       -

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# DEFAULT_SECURITY
# Format is:
#  securityPosture password privacy
#-------------------------------------------------------------------------------------------------------------------------------------------------------
DEFAULT_SECURITY semi-secure defaultpassword no

#-------------------------------------------------------------------------------------------------------------------------------------------------------
# Any SNMP agent configuration entries added by dynamic configuration
# (SET) requests get added to the end of the SNMPD.CONF file.
#-------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figure 39. SNMPD.CONF File - Part 3 of 3

The sample OSNMP.CONF file used by the osnmp command contains entries that match the sample SNMPD.CONF data set. See “Configure the OS/390 UNIX SNMP (osnmp) Command” on page 861 for additional information on configuring the osnmp command.

SNMPD.BOOTS

The SNMP agent uses the SNMPD.BOOTS configuration file to support SNMPv3 security. This file contains agent information used to authenticate the SNMPv3 requests. The SNMPD.BOOTS keeps the agent identifier and the number of times the agent reboots. If no SNMPD.BOOTS file exists when the agent is started, the agent creates one. You may want to add comments to the beginning of this file. If a file does exist, the agent uses the values specified in the file for setting its engineID and engineBoots values. If the file exists but contains invalid values for engineID or engineBoots, the agent issues a message and terminates.

Note: The recommended approach is to allow the SNMP agent to create the file.

SNMPD.BOOTS Search Order

To access SNMPD.BOOTS information, the search order is:

1. The name of an HFS file or an MVS file specified by the SNMPD_BOOTS environment variable.
2. /etc/snmpd.boots

Note: If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engineIDs are used for different SNMP agents.

SNMPD.BOOTS Statement Syntax

The syntax is:

engineID engineBoots

where:

engineID
A string of 2–64 (must be an even number) hexadecimal digits. The engine identifier uniquely identifies the agent within an administrative domain. By default, the engine identifier is created using a vendor-specific formula and incorporates the IP address of the agent. However, a customer can choose to use any engine identifier that is consistent with the snmpEngineID definition in RFC 2271 and that is also unique within the administrative domain.

engineBoots
The number of times (in decimal) the agent has been restarted since the engineID was last changed.

Notes:

1. engineID and engineBoots must be specified in order and on the same line.
2. Comments are specified in the file by starting the line with either an asterisk (*) or a pound sign (#).
3. No comments are allowed between the engineID and engineBoots values.
4. Only the first non-comment line is read. Subsequent lines are ignored.

Creating User Keys

Authentication

Authentication is generally required for SNMPv3 requests to be processed (unless the security level requested is 'noAuth'). When authenticating a request, the SNMP agent verifies that the authentication key sent in an SNMPv3 request can be used to create a message digest that matches the message digest created from the authentication key defined for the user.

The osnmp command uses the authentication key found on an entry in the OSNMP.CONF configuration file. It needs to correlate with the authentication key specified on a USM_USER entry for that user in the agent's SNMPD.CONF configuration file.

As an alternative to storing authentication keys in the client configuration file, the osnmp command allows user passwords to be stored. If the osnmp command is configured with a password, the code generates an authentication key (and privacy key if requested) for the user. These keys must, of course, produce the same authentication values as the keys configured for the USM_USER in the agent's SNMPD.CONF file or configured dynamically with SNMP SET commands. However, the use of passwords in the client configuration file is considered less secure than the use of keys in the configuration file.

The authentication key is generated from two pieces of information:

- The specified password
- The identification of the SNMP agent at which the key will be used. If the agent is an IBM agent and its engineID was generated using the vendor-specific engineID formula, the agent may be identified by IP address or host name. Otherwise, the engineID must be provided as the agent identification.

A key that incorporates the identification of the agent at which it will be used is called a localized key. It can be used only at that agent. A key that does not incorporate the engineID of the agent at which it will be used is called non-localized.

Keys stored in the osnmp command's configuration file, OSNMP.CONF, are expected to be non-localized keys. Keys stored in the SNMP agent's configuration file, SNMPD.CONF, can be either localized or non-localized, though the use of localized keys is considered more secure.

Encryption

Encryption is supported as a separately-orderable product, where export laws allow. Keys used for encryption are generated using the same algorithms as those used for authentication. However, key lengths may differ. For example, an HMAC-SHA authentication key is 20 bytes long, but a localized encryption key used with HMAC-SHA is only 16 bytes long.

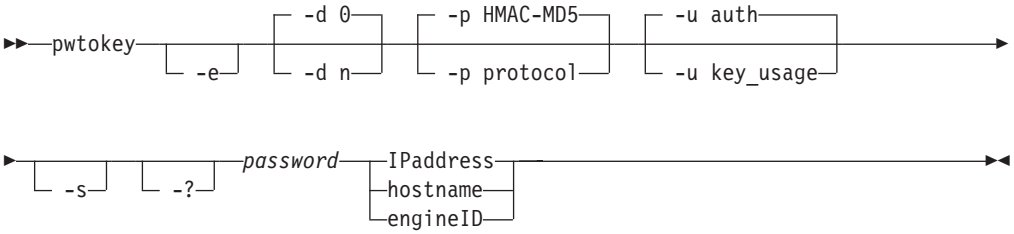
Using the pwtokey Facility

CS for OS/390 provides a facility called pwtokey that enables conversion of passwords into localized and non-localized authentication and privacy keys. The pwtokey procedure takes as input a password and an identifier of the agent and generates authentication and privacy keys. Since the procedure used by the pwtokey facility is the same algorithm used by the osnmp command, the person

configuring the SNMP agent can generate appropriate authentication and privacy keys to put in the SNMPD.CONF file for a user, given a particular password and the IP address at which the agent will run.

To convert passwords into authentication and privacy keys, issue the following command from OS/390 UNIX to use the pwtokey facility.

Syntax:



Parameters:

-e This flag indicates that the agent for which the key is being defined is identified by engineID rather than by IP address or host name. The engineID must be a string of 1–32 octets (2 to 64 hex digits). The default is that the agent identification is not an engineID.

-d This flag indicates what level of debug information is desired. Debug tracing is either on or off, so a value of 1 will cause debug tracing to be generated to the screen of the command issuer (sysout), and a value of 0 specifies that no debug tracing is to be generated. Debug tracing is off (0) by default.

-p This flag indicates for which protocols the keys should be generated. Valid values are:

HMAC-MD5
Generates keys for use with the HMAC-MD5 authentication protocol.

HMAC-SHA
Generates keys for use with the HMAC-SHA authentication protocol.

all Generate both HMAC-MD5 and HMAC-SHA keys.

The default is HMAC-MD5.

-u This flag indicates the usage intended for the key. Valid values are:

auth An authentication key

priv A privacy key

all Both authentication and privacy keys

Note: There is no difference between a key generated for authentication and a key generated for privacy. However, the length of privacy keys depends on whether the key is localized or not.

-s This flag indicates that output data should be displayed with additional spaces to improve readability. By default, data is displayed in a condensed format to facilitate cut-and-paste operations on the keys into configuration files or command lines.

-? This flag is used to invoke the pwtokey usage statement.

password
Specifies the text string to be used in generating the keys. The *password* must be 8–255 characters long. In general, while any printable characters may be used in the passwords, the OS/390 UNIX shell may interpret some characters rather than passing them to the pwtokey command. Include passwords in single quotes to avoid interpretation of the characters by the OS/390 UNIX shell.

Note: This password is not related to the community name (or ″password″) used with community-based security (SNMPv1 and SNMPv2c). This password is used only to generate keys for user-based security, an entirely different security scheme.

ipaddress
Specifies the IP address in dotted decimal notation of the SNMP agent at which the key will be used on an SNMP request.

hostname
Specifies the SNMP agent at which the key will be used on an SNMP request.

engineID
Specifies the engineID of the SNMP agent at which the key will be used. The engineID is determined at SNMP agent initialization from the SNMPD.BOOTS file.

Usage Notes: If the IP address or the host name is specified, the SNMP agent must be an IBM agent. The engineID is created using a vendor-specific formula that incorporates the IP address of the agent and an enterprise ID representing IBM.

Keys can be changed dynamically using the pwchange command. For information on the pwchange command, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

Provide MIB Object Configuration Information

An installation can set values for selected MIB objects by providing OSNMPD.DATA information. If no OSNMPD.DATA file is found, the defaults for these MIB objects are as follows:

Object	Default
dpiPathNameForUnixStream	The default is /tmp/dpi_socket.
sysDescr	If the environment variable HOSTNAME exists, its value is used. Otherwise, the default value identifies the OS/390 system under which the agent is running. The maximum length of this object is 255 octets.
sysContact	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysLocation	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysName	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysObjectId	1.3.6.1.4.1.2.3.13
sysServices	A single octet that defaults to 0. See the RFC 1907 description for this object.
snmpEnableAuthenTraps	Default value is 2, which means traps are disabled.
saDefaultTimeout	5 seconds
saMaxTimeOut	600 seconds
saAllowDuplicateIDs	Default is 1, which means multiple instances of a subagent are allowed. **Note:** Because a subagent identifier cannot be specified for DPI version 1 subagents, a constant identifier is used for all version 1 subagents. Therefore, this object must be set to ″Yes″ (1) to allow multiple DPI version 1 subagents to run concurrently.

For information about where these MIB objects are defined, refer to the *OS/390 SecureWay Communications Server: IP User's Guide*.

OSNMPD.DATA Search Order

To access the OSNMPD.DATA information, the search order is:

1. The name of an HFS file or MVS file specified by the OSNMPD_DATA environment variable.
2. /etc/osnmpd.data HFS file
3. The data set specified on the OSNMPD DD statement in the agent procedure
4. *jobname*.OSNMPD.DATA, where *jobname* is the name of the job used to start the SNMP agent

5. SYS1.TCPPARMS(OSNMPD)
6. *hlq*.OSNMPD.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used.

The first file found in the search order will be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

OSNMPD.DATA Statement Syntax

The OSNMPD.DATA statements specify MIB objects and their values. The format of each statement is:

```
object_name   value
```

The following rules apply:

- There can only be one *object_name* and associated *value* per statement.
- The *value* (if the *value* is a display or octet string) is case-sensitive and is saved in mixed case.
- Any display or octet string *value* that has imbedded white space must be enclosed in double quotes. For example, see the sysName entry in "OSNMPD.DATA Example".
- If the *value* is a display or octet string, it must be enclosed within double quotes.
- An entry must be contained on one line.
- Sequence numbers are not allowed on the statements.
- Comments begin with either an asterisk (*) or a pound sign (#).

OSNMPD.DATA Example

A sample of OSNMPD.DATA is installed as HFS file /usr/lpp/tcpip/samples/osnmpd.data. This sample can be modified for your installation.

```
#
# osnmpd.data sample
#
# Sample file for setting MIB variables and options for
# the SNMPv3 Agent provided by OS/390
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5647-A01
# (C) Copyright IBM Corp. 1996, 1998
# Status = CSV2R7
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "Unknown"
  sysLocation "Unknown"
  sysName "CS for OS/390 V2R7"
  sysObjectID     "1.3.6.1.4.1.2.3.13"
  snmpEnableAuthenTraps 1
  saDefaultTimeout    6
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to enable multiple DPI version 1
  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/tmp/dpi_socket"
```

```
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

Start the SNMP Agent (OSNMPD)

OSNMPD Parameters

The SNMP agent (OSNMPD) runs in a separate address space that executes load module EZASNMPD. OSNMPD can be started without parameters or you can add any of the parameters listed below.

When starting OSNMPD from MVS, add the parameters to the `PARMS=` keyword on the EXEC statement of the OSNMPD cataloged procedure. When starting OSNMPD from OS/390 UNIX, specify the desired parameters on the osnmpd command.

Note: The parameters are case sensitive. They must be entered in lower case.

Parameter	Description
-c *community*	Specifies a community name. The community name is a password that can accompany an SNMP request that the agent receives. Specifying a community on this parameter when starting the agent causes the community to effectively be added to the list of community names in PW.SRC, with a mask and IP address of zeros. Therefore, any request received with this *community* would be authenticated (for example, the request would be accepted from any IP address).
-d *level*	Specifies the level of tracing to be started. The valid values for *level* are 0–255. If the -d parameter is not specified, then the default level of 0 is used, meaning no tracing will be done. If the -d parameter is specified without a *level*, then a level of 31 is used, meaning all SNMP requests/responses/traps and DPI activity will be traced. There are 8 levels of tracing provided. Each level selected has a corresponding number. The sum of the numbers associated with each level of tracing selected is the value which should be specified as *level*. Once the agent is started, tracing options can be dynamically changed using the MVS MODIFY command. For more information on agent tracing, see the *OS/390 SecureWay Communications Server: IP Diagnosis*. The numbers for the trace levels are: **0** No tracing (default) **1** Trace SNMP requests **2** Trace SNMP responses **4** Trace SNMP traps **8** Trace DPI packets **16** Trace DPI internals (currently, no specific traces are recorded for this trace level) **32** Agent internal trace **64** Agent internal trace plus extended storage dump traces **128** Agent internal trace plus extended storage dump traces plus additional information

-i *interval* Specifies the interval (in minutes) at which dynamic configuration changes to the SNMP agent should be written out to the SNMPD.CONF configuration file. Valid values are 0–10. The default value is 5. This parameter is only relevant when the SNMPD.CONF file is used for SNMPv3 configuration.

-p *port* Listens for SNMP packets on this port. The default is port 161. If you change the port to something other than 161, you must also configure any subagents and commands, such as osnmp, to use the new port. For more information on defining another port other than 161 for SNMP requests, see "Provide TCP/IP Profile Statements" on page 826.

-s *socketname* Specifies the name of the UNIX socket to be used in accepting requests from subagents that communicate with the agent via AF_UNIX connections. This value can be configured either by specifying it on the `-s` parameter or by specifying it as the value of the `dpiPathNameForUnixStream` MIB object in OSNMPD.DATA. The default is `/tmp/dpi_socket`. The file permission bits for this file must be read and write for the subagents to connect.

? Displays the usage statement for the command. If this parameter is specified, all other parameters are ignored. If OSNMPD was started from MVS then the usage information is written to syslogd. If OSNMPD was started from OS/390 UNIX, then the usage information is displayed to the invoker of the command.

Sample JCL Procedure for Starting OSNMPD from MVS

Update cataloged procedure OSNMPD by copying the sample in *hlq*.SEZAINST(OSNMPDPR) to your system or recognized PROCLIB. Change the data set names as required to suit your local configuration. The OSNMPD address space requires access to the IBM C/370 Library during execution.

Parameters may be passed to the agent on the `PARMS=` keyword on the EXEC statement of the OSNMPD cataloged procedure. For a detailed explanation of the parameters, see "OSNMPD Parameters" on page 852. Any agent parameters you wish to specify may be added as shown in the following example:

```
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/ -c abc -d 255 -p 761'
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of commnity names supported by the agent, and all agent traces will be activated. For more information on tracing, see the *OS/390 SecureWay Communications Server: IP Diagnosis*.

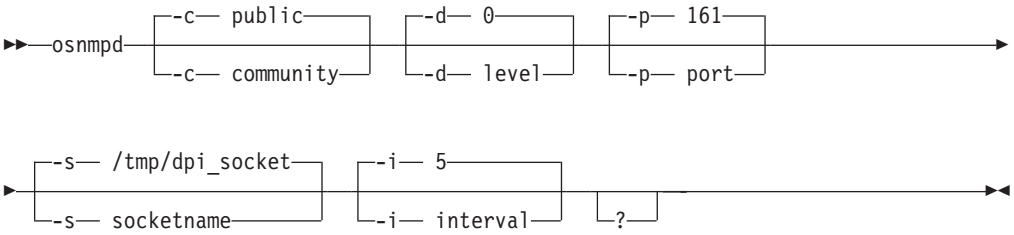
Following is the sample OSNMPD procedure:

```
//OSNMPD    PROC
//*
//* Sample procedure for running the OE SNMP agent
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: AEZASMP1
//*
//*         Licensed Materials - Property of IBM
//*         "Restricted Materials of IBM"
//*         5647-A01
//*         (C) Copyright IBM Corp. 1997, 1998
//*         Status = CSV2R7
```

```
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/-d 0'
//*
//* - The C runtime libraries should be in the system's link list
//*   or this sample procedure will need to STEPLIB to them.
//*
//* - TCP/IP runtime libraries should also be in the system's link
//*   list.
//*
//* - OSNMPD must find the name (TCPIPJOBNAME in TCPIP.DATA) that
//*   it should be associated with. The OE function __iptcpn() is
//*   used to find this name. It is suggested that the parmlist
//*   be modified to set the environment variable
//*   RESOLVER_CONFIG to point to the correct resolver file when
//*   multiple INET Physical File Systems are started:
//*
//*   //      PARM=('POSIX(ON) ALL31(ON)',
//*   //      'ENVAR("RESOLVER_CONFIG=/etc/tcpv33a.data")/-d 0')
//*
//*   To specify an MVS dataset, code:
//*
//*   //  PARM=('POSIX(ON) ALL31(ON)',
//*   //  'ENVAR("RESOLVER_CONFIG=//''TCPV33.MYFILE(TCPDATA)''")/-d 0')
//*
//*   If only one INET PFS will be started then it is
//*   recommended that /etc/resolv.conf be used.
//*
//* - The OSNMPD agent can also be invoked from the OMVS shell as
//*   a shell command.  An alternative way of invoking the OSNMPD
//*   agent from batch is to use the BPXBATCH facility to issue the
//*   osnmpd shell command.  This is a good method to use when the
//*   agent's configuration datasets are going to be identified by
//*   environment variable (OSNMPD_DATA, SNMPD_CONF, SNMPD_BOOTS).
//*   However, if configuration data sets are going to be identified
//*   using DD statements (//SYSPWSRC, //SNMPTRAP, //OSNMPD), this
//*   this sample JCL is easier.
//*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

Command for Starting OSNMPD from OS/390 UNIX

Following is the command syntax used to start OSNMPD from OS/390 UNIX.



Note: Each parameter must be separated by a blank. Parameter values can be specified in mixed case.

To run the SNMP agent in background, you must add an ampersand (&) to the command and the issuer of the command must be in OS/390 UNIX superuser mode. For a detailed explanation of the osnmpd parameters, see “OSNMPD Parameters” on page 852.

Any agent parameters you wish to specify may be added as shown in the following example:

```
osnmpd -c abc -d 255 -p 761
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of commnity names supported by the agent, and all agent traces will be activated. For more information on tracing, see *OS/390 SecureWay Communications Server: IP Diagnosis*.

Step 2: Configure the SNMP Commands

The two SNMP client applications provided with CS for OS/390 are:

- SNMP command from the NetView environment
- osnmp command in the OS/390 shell

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The osnmp command in the OS/390 shell supports SNMP versions 1, 2, and 3. Depending on your requirements, you may decide to configure either or both of these clients, or to use an SNMP client on another platform.

Configure the NetView SNMP (SNMP) Command

The SNMP command in the NetView environment can be used to send SNMP version 1 requests to SNMP agents on either local or remote hosts. The SNMP command requires the command processor itself, the SNMPIUCV task for inter-address space communication, and the SNMP query engine, which creates the packets sent to the SNMP agent. The NetView SNMP command supports only community-based security.

Configure the SNMP Query Engine

Update the SNMPQE cataloged procedure by copying the sample in *hlq*.SEZAINST(SNMPPROC) to your system or recognized PROCLIB. Specify SNMP parameters and change the data set names as required to suit your local configuration. The SNMPQE address space requires access to the IBM C/370 Library during execution.

The SNMP query engine (SQESERV) needs access to the *hlq*.MIBDESC.DATA data set for the MIB variable descriptions. You can find a sample of this data set in *hlq*.SEZAINST(MIBDESC).

MIBDESC.DATA Data Set: The MIBDESC.DATA data set 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 SNMP agent was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP on CS for OS/390 ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP

agent. 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 MIBDESC.DATA data set and uses the ASN.1 name specified in the data set when it sends the SNMP packet to the agent.

The SNMPQE address space must be able to access the MIBDESC.DATA data set.

You can change the short names in the MIBDESC.DATA data set 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 data set that satisfies the search. In addition, all enterprise-specific variables used by hosts in your network should be added to this data set.

Entries in the data set 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.

Following is a sample MIBDESC.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):

```
*-------------------------------------------------------------*
* MIB Variable name | ASN.1 notation        | Type     | TTL  *
*-------------------------------------------------------------*
* Following is Dutch name for sysDescr
systeemBeschrijving  1.3.6.1.2.1.1.1.         display    900
sysDescr             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
```

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. In the previous example, the SNMP Query Engine would return `systeemBeschrijving` and not `sysDescr`. The name returned is in mixed case.

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

MIBDESC.DATA data set. 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 MIBDESC.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.

Note: If you are using SNMP to receive response or trap PDUs which contain enterprise specific variables, the variables must be added to the MIBDESC.DATA data set.

SNMPQE Cataloged Procedure (SNMPPROC):

```
//SNMPQE   PROC MODULE=SQESERV,PARMS=''
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEB01W
//*
//*        5655-HAL (C) Copyright IBM Corp. 1989, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//SNMPQE   EXEC PGM=&MODULE,PARM='&PARMS',
//             REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the SETPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZADSIL,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        MSSNMPMS identifies an optional data set for NLS support.
//*        It specifies the SNMP message repository.
//*
//*MSSNMPMS DD DSN=TCPIP.SEZAINST(MSSNMP),DISP=SHR
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Specifying the SNMPQE Parameters: The SQESERV module can be configured to start without parameters or you can add any of the following parameters to PARMS=' in the PROC statement of the SNMPQE cataloged procedure. For example,

```
//SNMPQE   PROC MODULE=SNMPQE,PARMS='-h MVSA'
```

Notes:

1. These parameters are also valid when starting SNMPQE with the START command.
2. The commands are case sensitive. They must be entered in lower case.

Parameter	Description
-d *trace_level*	Specifies the level of tracing to be run. Valid values for the trace level are: **0** No tracing (default) **1** Displays errors **2** In addition to errors, also displays SNMP query engine protocol packets sent and received **3** In addition to 2, also displays the SNMP packets sent and received **4** In addition to 3, also displays the buffers in hexadecimal format
-h *host_name*	Specifies the IP address to which to bind, so that SQESERV accepts connections only 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 be done. This is only used for debugging the socket layer in a user's application. It can result in a large amount of STDOUT output.

See *OS/390 SecureWay Communications Server: IP Diagnosis* for more information on tracing.

Setting Up Authorization for SNMPQE: To create RAW sockets necessary for SNMP PING requests, the user ID associated with the SNMPQE started task must have superuser authority (OS/390 UNIX UID of 0) or be permitted to BPX.SUPERUSER facility authority.

Configure NetView as an SNMP Monitor

To configure the NetView interface as an SNMP monitor, perform each of the following tasks:

- Configure for SNMPIUCV
- Configure for the SNMP command processor
- Configure for the SNMP messages
- Update the SNMP initialization parameters

Configure for SNMPIUCV: SNMPIUCV is the NetView optional task that handles IUCV communication between the NetView program and the SNMP query engine. SNMPIUCV resides in the *hlq*.SEZADSIL data set.

Add the following TASK statement for SNMPIUCV to the DSIDMN member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

This statement causes SNMPIUCV to start automatically when the NetView program is started.

If you specify `INIT=N` instead of `INIT=Y` 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 by the SNMPQERT keyword in the SNMPARMS member of the *hlq*.SEZADSIP data set. The default is every 60 seconds.

Configure for the SNMP Command Processor: SNMP is the command processor that allows NetView operators and CLISTs to issue SNMP commands. SNMP resides in the *hlq*.SEZADSIL data set. This data set should be concatenated to the STEPLIB DD statement in the NetView start procedure.

Add the following statement to the DSICMD member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=Y
```

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.

Configure for the SNMP Messages: The NetView SNMP messages reside in the *hlq*.SEZADSIM data set as DSISNM*nn*, where *nn* is the number of the member. The valid message members are DSISNM00 through DSISNM05, DSISNM10, DSISNM12, and DSISNM99. The data set containing these members should be added to the DSIMSG DD statement in the NetView start procedure.

Update the SNMP Initialization Parameters: SNMPIUCV reads the SNMPARMS member in the *hlq*.SEZADSIP data set at startup. This data set contains the initialization parameters for SNMP. The data set containing SNMPARMS should be added to the DSIPARM DD statement in the NetView startup procedure.

Sample SNMP Parameter Data Set (SNMPARMS): Following is the SNMPARMS sample:

```
*-------------------------------------------------------------*
* Member name:                                                *
*      SNMPARMS                                               *
*                                                             *
* Copyright:                                                  *
*   Licensed Materials - Property of IBM                      *
*   This product contains "Restricted Materials of IBM"       *
*   5655-HAL (C) Copyright IBM Corp. 1990, 1994.              *
*   All rights reserved.                                      *
*   US Government Users Restricted Rights -                   *
*   Use, duplication or disclosure restricted by GSA ADP      *
*   Schedule Contract with IBM Corp.                          *
*   See IBM Copyright Instructions.                           *
*                                                             *
* Function:                                                   *
*      Externalized paramters for SNMPIUCV module, the task   *
*      that communicates with the SNMP Query Engine.          *
*                                                             *
* Attributes:                                                 *
*      Read by DSIDKS to obtain actual parameters.            *
*                                                             *
```

```
* Library:                                                      *
*      On MVS:   Member SNMPARMS in NetView's DSIPRM dataset. *
*      On VM:    File SNMPARMS NCCFLST on a NetView minidisk. *
*                                                             *
* Change activity:                                            *
*                                                             *
*   PTM      DATE    DESCRIPTION                              *
* -------  -------   -----------------------------------------*
*          24Sep89   Initial version                          *
*          09Nov89   Final version                            *
*          20Feb90   Adapt defaults as recommended during test*
*          27Mar90   Adapt defaults for new AF_IUCV sockets   *
*          09Jul90   Remove not needed paramters              *
*                                                             *
*-------------------------------------------------------------*
*
  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
```

Specifying the SNMPARMS Parameters: You can change the following parameters in SNMPARMS:

Parameter **Description**

SNMPQE *name*

The name of the SNMP query engine started procedure. This value is case-sensitive. The default address space name is SNMPQE. If you change the name of the SNMP query engine started procedure, you must change this parameter to match the new procedure name.

SNMPQERT *seconds*

The retry timer, in 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 0–9999. The default is 60 seconds.

SNMPRCNT *number*

The retry count for sending SNMP requests. This is the number of times the SNMP query engine will resend an SNMP PDU when no response was received. If no response is received after all retries have been exhausted, the SNMP query engine will return a no response error for the SNMP request. The valid range of values is 0–255. The default is 2.

If the request being sent by the SNMP query engine contains an invalid community name, no response will be received. This causes the SNMP query engine to resend the request until the retry count is exhausted. If authentication failure traps are enabled, the agent generates multiple ***authentication-Failure*** traps, one for the initial request and one for each of the retries.

SNMPMMLL *length*

The line length for multi-line messages 38 and 44. The maximum length is 255. A value of 80 allows the complete text to appear on an 80-character-wide screen. The default and minimum acceptable line length value is 80.

SNMPRETO *exp*

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 1-second intervals until all retries have been sent. SNMPRETO of 2 causes the first retry to be sent after 1 second, the second retry 2 seconds later, the third retry 4 seconds later, and so on until all retries have been sent.

SNMPRITO *tenths_seconds*

The time-out value for a request specified in tenths of a second. After sending an SNMP request to an agent, the SNMP query engine waits the specified number of tenths of a second for a response.

- If the retry count (SNMPRCNT) is greater than 0, the SNMP request is sent again if a response is not received in this time.
- If the retry count (SNMPRCNT) is 0, a `no response` error is sent to the NetView program, if a response is not received within this period of time.

The valid range of values is 0–255. The default is 10 tenths of a second.

Configure the OS/390 UNIX SNMP (osnmp) Command

The osnmp command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2, or SNMPv3. For SNMPv2 and SNMPv3 requests, the OSNMP.CONF configuration file is required. The *winSNMPname* specified on an OSNMP.CONF statement can be used as the value of the -h parameter on the osnmp command. For a detailed explanation of the parameters you can specify on the osnmp command, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

To configure the osnmp command, perform the following tasks:

- Provide osnmp configuration information
- Provide user MIB object information

Provide osnmp Configuration Information

The OSNMP.CONF file is used to define target agents and, for SNMPv3, the security parameters to be used in sending requests to them. The following search order for this file enables different copies of the file to be used by different users:

1. An HFS file or MVS data set pointed to by the OSNMP_CONF environment variable
2. /etc/osnmp.conf
3. /etc/snmpv2.conf

The contents of the file, regardless of location, is the same. Only the first file found is used. A sample of this file is installed as HFS file /usr/lpp/tcpip/samples/snmpv2.conf. This sample, shown below, should be copied over to the /etc directory and modified for your installation.

```
# osnmp.conf sample
#   (used as /etc/snmpv2.conf unless OSNMP_CONF environment variable set)
#
# Sample file showing format of configuration file for the osnmp command
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5647-A01
# (C) Copyright IBM Corp. 1996, 1998
# Status = CSV2R7
#
#-------------------------------------------------------------------------------------------------------------------------------------------------------
#
# Format of entries:
#   winSNMPname  targetAgent  admin  secName  password  context secLevel authProto  authKey  privProto  privKey
#
#-------------------------------------------------------------------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------------------------------------------------------------------
# Community-based security  (SNMPv1 and SNMPv2c)
#-------------------------------------------------------------------------------------------------------------------------------------------------------
v1       127.0.0.1   snmpv1
v2c      127.0.0.1   snmpv2c
mvs1     9.67.113.79 snmpv2c
mvs2     mvs2c       snmpv2c
mvs3     mvs3:1061   snmpv2c
#
#-------------------------------------------------------------------------------------------------------------------------------------------------------
# User-based Security Model (USM with SNMPV3)
#
# Notes
# - Keys in this file must not be localized.
# - All keys need to be regenerated using the pwtokey command in order
#   for these sample entries to actually be used.
# - In this sample:
#   - Keys are generated for use with engineID 00000002000000000943714F
#   - Authentication keys were generated with password of
#     username+"password", such as "u1password"
#   - Privacy keys were generated with password of
#     username+"privpass", such as "u1privpass"
#   - Entries defined to use encryption support, which is available only
#     as a separately orderable feature on the base OS/390 product, are
#     included below but commented out.
#
#-------------------------------------------------------------------------------------------------------------------------------------------------------
#
#v3mpk  127.0.0.1 snmpv3 u1 -          - AuthPriv     HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987         DES eac02a0d9fe90eca7911fdcaba20deae
v3mak   127.0.0.1 snmpv3 u1 -          - AuthNoPriv   HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987         -   -
v3n     127.0.0.1 snmpv3 u1 -          - noAuthNoPriv -        -                                        -   -
#
#v3spk  127.0.0.1 snmpv3 u2 -          - AuthPriv     HMAC-SHA 76784e5935acd6033a855df1fac42acb187aa867 DES adaaf313277a55a3df3a8d2fb70192c427799e0c
v3sak   127.0.0.1 snmpv3 u2 -          - AuthNoPriv   HMAC-SHA 76784e5935acd6033a855df1fac42acb187aa867 -   -
#
#v3mpk2 127.0.0.1 snmpv3 u3 -          - AuthPriv     HMAC-MD5 d1f86e9c9346253c10a4cd2da339b1db         DES cfccde3249ba521ae4da0ddfc2b76ee7
#
#v3spk2 127.0.0.1 snmpv3 u4 -          - AuthPriv     HMAC-SHA 42529b3c6c138c173e70db1050de8d74c04205cb DES ef70a0a98d399a9189f3169e82010f3b46e694e2
#
#v3mpp  127.0.0.1 snmpv3 u5 u5password - AuthPriv     HMAC-MD5 -                                       DES -
v3map   127.0.0.1 snmpv3 u5 u5password - AuthNoPriv   HMAC-MD5 -                                       -   -
#
#v3spp  127.0.0.1 snmpv3 u6 u6password - AuthPriv     HMAC-SHA -                                       DES -
v3sap   127.0.0.1 snmpv3 u6 u6password - AuthNoPriv   HMAC-SHA -                                       -   -
```

OSNMP.CONF Statement Syntax: The configuration file is required when sending requests to the SNMPv2 or SNMPv3 nodes in your network. The configuration file can also be used to send SNMPv1 requests.

The syntax of a statement in the configuration file is:

winSNMPname targetAgent admin secName password context secLevel authProto authKey privProto privKey

Field definitions:

winSNMPname
An administrative name by which the winSNMP code used by osnmp can locate an entry in this configuration file. There is no default value. Specified on the `-h` option. (Maximum 32 characters.)

targetAgent
Host name or IP address of the node of the target agent (maximum 80 characters). There is no default value. To direct the command to a port other than 161, specify *host:port#*. For example, for port 222 at mvs150, specify mvs150:222. Port number, if specified, must be between 1 and 65535.

admin Specifies the administrative model supported by the targetAgent. Valid values are:

- 'snmpv1' - Community-based SNMPV1 security
- 'snmpv2c' - Community-based SNMPV2 security
- 'snmpv3' - User-based SNMPV3 security

There is no default value.

secName
Specifies the security name of the principal using this configuration file entry. For user-based security, this is the userName. The user must be defined at the targetAgent. This field is ignored unless 'snmpv3' is specified for the admin keyword. A valid value is a user name of 1–32 characters. There is no default.

password
Specifies the password to be used in generating the authentication and privacy keys for this user. If a password is specified, it is used to automatically generate any needed keys and the ″authKey″ and ″privKey″ fields below are ignored. This field is ignored unless 'snmpv3' is specified for the admin keyword. If no password is desired, set field to a single '-' (dash). (Minimum 8 characters, maximum 64 characters.)

Note: The use of password instead of keys in this configuration file is not recommended, as storing passwords in this file is less secure than using keys.

context
The SNMP contextName to be used at the target agent. The contextName is needed only at agents that support multiple contexts; otherwise, the only context supported is the null context, which is the default value of this keyword. The CS for OS/390 SNMP agent does not support multiple contexts. This field is ignored unless 'snmpv3' is specified for the admin keyword. If the blank ″″ context selector is desired, set this field to a single '-' (dash). (Maximum 32 characters).

secLevel
Specifies the security level to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless 'snmpv3' is specified for the admin keyword. Privacy is supported in CS for OS/390 only in a separately orderable product. It is not supported in the base product. Valid values are 'noAuthNoPriv' or 'none' to indicate that no authentication or privacy is requested, 'AuthNoPriv' or 'auth' to indicate that authentication is requested but privacy is not requested, 'AuthPriv' or 'priv' to indicate that both authentication and privacy are requested (only supported in the additional encryption product), or a '-' (dash) to indicate the default value (noAuthNoPriv).

authProto
SNMP authentication protocol to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless 'snmpv3' is specified for the admin keyword. Valid values are 'HMAC-MD5', 'HMAC-SHA', or a single '-' (dash) for no authentication.

authKey
Specifies the SNMP authentication key to be used in communicating with the target SNMP agent when this entry is used. This key must be the non-localized key. This field is ignored if the password keyword is used. This field is ignored unless 'snmpv3' is specified for the admin keyword and a non-default value is specified for authProto. Valid values are 16 bytes (32

hex digits) when authProto is 'HMAC-MD5' and 20 bytes (40 hex digits) when authProto is 'HMAC-SHA'. A '-' (dash) indicates the default value, which is no key.

privProto
Specifies the SNMP privacy protocol to be used in communicating with the target SNMP agent when this entry is used. Privacy is supported on CS for OS/390 only in a separately orderable product. It is not supported in the base product. If privacy is not supported, this keyword is ignored. This field is ignored unless 'snmpv3' is specified for the admin keyword. Valid values are 'DES' for CBC-DES (only supported in the additional encryption product) or a '-' (dash) to indicate the default value, which is no privacy.

privKey
Specifies the SNMP privacy key to be used in communicating with the target SNMP agent when this entry is used. This key must be the non-localized key. This field is ignored if the password keyword is used. If privacy is not supported, this keyword is ignored. The privacy and authentication keys are assumed to have been generated using the same authentication protocol (for example, both with HMAC-MD5 or both with HMAC-SHA). This field is ignored unless 'snmpv3' is specified for the admin keyword and a non-default value is specified for privProto. Valid values are 16 bytes (32 hex digits) when authProto is 'HMAC-MD5', 20 bytes (40 hex digits) when authProto is 'HMAC-SHA', or a '-' (dash) to indicate the default value (no key).

- All parameters for an entry must be contained on one line in the configuration file.
- A "-" (dash) indicates the default value for a keyword.
- Sequence numbers are not allowed on the statements.
- Comments begin with a pound sign (#) in column 1.
- The secName and password parameters are case-sensitive.
- The pwtokey command can be used to generate the authentication and privacy keys. For information on the pwtokey command, see "Creating User Keys" on page 847.
- Because the osnmp command supports both issuance of SNMP requests and receipt of SNMP traps, the entries in the OSNMP.CONF file must be defined for both uses. Multiple entries for the same USM user are allowed within the file. This may be useful when defining different security levels for the same user. If multiple entries for the same USM user are defined, be aware that only the first one in the file can be used for receiving notifications. If multiple entries for the same USM user are defined and the user will receive notifications, the definition with the highest (most stringent) securityLevel should be defined first. Doing so will allow the user to be used for any level of security equal to or lower (less stringent) than the securityLevel defined.

Examples:

- Example 1:

  The following entry defines an SNMPv2c node for osnmp:

  ```
  mvs1     9.67.113.79 snmpv2c
  ```

  where *mvs1* is the name used with the -h parameter on the osnmp command and *9.67.113.79* is the IP address of the SNMPv2c agent.
- Example 2:

The following entry defines an SNMPv3 node:
where *v3mak* is the name used on the -h parameter of the osnmp command.

```
v3mak   127.0.0.1 snmpv3 u1 - - AuthNoPriv  HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987 - -
```

SNMP requests sent using this entry uses USM user name *u1* using HMAC-MD5 authentication but no encryption.

- Example 3:

  The following entry defines an SNMPv3 node. The needed authentication and privacy keys will be generated from the password *u6password*.
  The USM user is *u6*. The authentication protocol is HMAC-SHA, and CBC 56-bit

```
v3sap   127.0.0.1 snmpv3 u6 u6password - AuthNoPriv   HMAC-SHA -   DES -
```

DES encryption is used.

Provide User MIB Object Information

If you want to use the textual names for MIB objects which are not defined in any compiled MIB, then you can define them to the osnmp command in an /etc/mibs.data HFS file. All of the objects in the list in the “Management Information Base (MIB) Objects” appendix in the *OS/390 SecureWay Communications Server: IP User’s Guide.* are in the compiled MIB. A sample of the /etc/mibs.data file is installed as HFS file /usr/lpp/tcpip/samples/mibs.data. This sample, shown below, should be copied over to the /etc directory and modified for your installation.

```
#      Licensed Materials - Property of IBM
#      This product contains "Restricted Materials of IBM"
#      5647-A01 (C) Copyright IBM Corp. 1996, 1997
#      All rights reserved.
#      US Government Users Restricted Rights -
#      Use, duplication or disclosure restricted by
#      GSA ADP Schedule Contract with IBM Corp.
#      See IBM Copyright Instructions.
# short name            OID                  type
#-------------------------------------------------

# my system variables

myDescr              1.3.6.1.2.1.1.1.          display
myObjectid           1.3.6.1.2.1.1.2.          objectidentifier
myUptime             1.3.6.1.2.1.1.3.          Timeticks
myContact            1.3.6.1.2.1.1.4.          display
myName               1.3.6.1.2.1.1.5.          display
myLocation           1.3.6.1.2.1.1.6.          display
myServices           1.3.6.1.2.1.1.7.          integer

# DPI SAMPLE MIB (DPI version 2.0 sample program)

dpiSimpleInteger    1.3.6.1.4.1.2.2.1.5.1.0   integer
dpiSimpleString     1.3.6.1.4.1.2.2.1.5.2.0   display
dpiSimpleCounter32  1.3.6.1.4.1.2.2.1.5.3.0   counter32
dpiSimpleCounter64  1.3.6.1.4.1.2.2.1.5.4.0   counter64

# DPI SAMPLE MIB (DPI version 1.1 sample program)

dpiSample              1.3.6.1.4.1.2.2.1.4.      null
dpiSampleNumber        1.3.6.1.4.1.2.2.1.4.1.    uinteger
dpiSampleNumberString  1.3.6.1.4.1.2.2.1.4.1.1.  display
dpiSampleOctetString   1.3.6.1.4.1.2.2.1.4.2.    octetstring
dpiSampleObjectID      1.3.6.1.4.1.2.2.1.4.3.    objectidentifier
dpiSampleEmpty         1.3.6.1.4.1.2.2.1.4.4.    display
dpiSampleInetAddress   1.3.6.1.4.1.2.2.1.4.5.    ipaddress
dpiSampleCounter       1.3.6.1.4.1.2.2.1.4.6.    counter
dpiSampleGauge         1.3.6.1.4.1.2.2.1.4.7.    gauge
```

```
dpiSampleTimeTicks     1.3.6.1.4.1.2.2.1.4.8.    timeticks
dpiSampleDisplayString 1.3.6.1.4.1.2.2.1.4.9.    displaystring
dpiSampleCommand       1.3.6.1.4.1.2.2.1.4.10.   displaystring
```

/etc/mibs.data Statement Syntax: The /etc/mibs.data statements can be used to specify character (usually called 'textual') names for MIB objects not defined in any compiled MIB supplied with CS for OS/390. You can then use these character/textual names as the name of the objects on the osnmp command.

The format of a statement in this file is:

character_object_name object_identifier object_type

Field definitions:

character_object_name
The character/textual name of the MIB object. The *character_object_name* value can contain both uppercase and lowercase letters; however, it is *not* case-sensitive.

object_identifier
The ASN.1 value for the MIB object.

object_type
The SMI_type for the MIB object. The valid SMI_type values are:
- bitstring
- counter
- counter32
- counter64
- dateAndTime
- display or display string
- integer
- integer32
- ipaddress
- gauge
- gauge32
- nsapaddress
- null
- objectidentifier or OID
- octetstring
- opaque
- opaqueascii
- snmpAdminString
- timeticks
- uinteger

- The maximum length of each statement in this file is 2048 bytes.
- All parameters for each character or textual name must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with a pound sign (#) in column 1.

Step 3: Configure the SNMP Subagents

This section contains information about configuring the TCP/IP subagent.

There are three SNMP subagents shipped with CS for OS/390:
- The TCP/IP subagent reports information about the TCP/IP stack.

- The OMPROUTE subagent reports information specific to OSPF. For more information, see “ROUTESA_CONFIG Statement” on page 991. For configuration information about the SLA subagent, see “Chapter 26. Configuring OMPROUTE” on page 949.
- The SLA subagent reports information about defined service policies and performance statistics related to traffic using those policies. For configuration information about the SLA subagent, see “Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent” on page 1095.

There are two statements in the profile data set used to configure the TCP/IP subagent, the SACONFIG and ITRACE statements.

- SACONFIG

  Use the SACONFIG statement to configure the subagent. The SACONFIG parameters determine whether or not the subagent is automatically started at TCP/IP initialization, what port number to use to contact the agent, and other configuration values. For a detailed explanation of this statement, see “SACONFIG Statement” on page 236.
- ITRACE

  Use the ITRACE statement to determine what trace information, if any, should be recorded by the subagent. For a detailed explanation of this statement, see “ITRACE Statement” on page 220.

The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, see “ROUTESA_CONFIG Statement” on page 991.

Step 4: Configure the ATM Open Systems Adapter 2 (ATM OSA-2) Support

The TCP/IP subagent can retrieve ATM OSA-2 data from the Open Systems Adapter Support Facility (OSA/SF) for support of ATM management. The ATM management data is organized into MIB tables defined in the IBM MVS TCP/IP Enterprise Specific MIB. The ATM management data is available for any ATM OSA-2 adapter cards, whether or not they are being used by the TCP/IP instance to which they are defined. As long as they are defined by DEVICE and LINK profile statements, SNMP can retrieve ATM management data for them even if they are defined to VTAM, a different TCP/IP instance, or if they are configured for ATM LAN emulation mode and are defined to this TCP/IP instance or another TCP/IP instance in the same MVS image as an LCS device.

Current support consists of:

- Interface table from RFC2233 for ATM devices, Native and IP Forwarding links, LAN emulation links, AAL5 and ATM layer interfaces
- ATM Channel, Port and PVC tables from the IBM MVS Enterprise Specfic MIB
- ATM LAN Emulation tables from the IBM MVS Enterprise Specific MIB
- atmInterfaceConfTable from RFC1695
- Asynchronous SNMP Trap generation for operational management:
  - ATM Port enabled - LinkUp Trap
  - ATM Port disabled - LinkDown Trap
  - Permanent Virtual Circuit (PVC) creation - ibmMvsAtmOsasfAtmPvcCreate Trap

  - Permanent Virtual Circuit (PVC) deletion - ibmMvsAtmOsasfAtmPvcDelete Trap
- Provide method for assigning an IP Address to the ATM Port

  Use the *osnmp set* command against the ibmMvsAtmOsasfPortIpAddress MIB object to assign the IP Address.

OSA/SF Prerequisites

The TCP/IP subagent provided by CS for OS/390 will connect to OSA/SF to provide for ATM Management. For a subagent to establish a connection to OSA/SF, two OSA/SF components must be started:

- IOAOSASF

  IOAOSASF is a sample JCL procedure that can be used to start the main OSA/SF address space. The sample has a jobname of OSASF1.
- IOASNMP

  IOASNMP is a sample JCL procedure that starts the OSA/SF-provided OS/390 UNIX transport application that interconnects a subagent with OSASF1.

These sample procedures and all entities that they call are provided with OSA/SF. For a detailed explanation of how to set up OSA/SF on your MVS system, see *Planning for the System/390 Open Systems Adapter Feature*. The primary purpose of OSA/SF is to manage OSA Adapters. It has been extended to support ATM Management via SNMP. An instance of IOAOSASF, IOASNMP, TCPIP and its subagent, and an SNMP agent must be started on every MVS image where ATM Management support is needed.

The recommended startup order is:

1. Start IOAOSASF and wait until it completely initializes. IOAOSASF must be started before IOASNMP.
2. Start TCP/IP and wait until TCP/IP completes its initialization. Actually IOAOSASF can be started after TCP/IP but must be started prior to the next step.
3. Start IOASNMP after TCP/IP is initialized. If starting multiple TCP/IP instances that run under OS/390 UNIX as AF_INET Physical File Systems, wait until at least one TCP/IP where OSA/SF support was requested has initialized. OSA/SF support is requested by specifying the OSASF parameter on the SACONFIG statement in the profile data set for a TCP/IP instance. For a detailed description of the SACONFIG statement, see “SACONFIG Statement” on page 236.
4. Start the SNMP Agent, OSNMPD, for each TCP/IP instance where ATM Management support is desired.

On an MVS image only a single instance of either IOAOSASF or IOASNMP can (or should) be started. An attempt to start multiple copies of IOAOSASF will be rejected. Starting multiple copies of IOASNMP will yield unpredictable results.

Ensure that OSA/SF is at Version 1 Release 2 level or higher with the OSA/SF APAR OW24712 applied. For SNMP ATM LAN emulation support, APAR OW24052 must be applied, and APAR OW28383 must be applied for SNMP ATM IP Forwarding support.

Required TCP/IP Profile Statements

For a detailed description of the statements mentioned here, see “Chapter 4. Configuring the TCP/IP Address Space” on page 97. The following TCP/IP profile statements must be updated for ATM Management support:

- SACONFIG

  On the SACONFIG statement, ATM Management support must be enabled by specifying ATMENABLED. Omission of ATMENABLED when TCP/IP is started will result in no ATM Management support. The SACONFIG statement controls the operation of the subagent that runs in a TCP/IP address space as a separate task.

  The OSASF parameter specifies which port IOASNMP should use to Listen for connections from subagents to OSA/SF. For an explanation of the usage of this parameter when starting multiple TCP/IP instances, see “Multiple TCP/IP Instances Considerations”. It is recommended that the OSASF port be reserved by also specifying it on a PORT statement.

  For example:

  ```
  SACONFIG ATMENABLED OSASF 721
  ```

- PORT

  Prereserve the port to be used in communication with OSA/SF.

  For example:

  ```
  PORT
     721  IOASNMP ; OSA/SF TCP/IP Communications
  ```

  In the above example since IOASNMP runs as an OS/390 UNIX application the port could have been reserved for OS/390 UNIX. Review the /etc/services HFS file to insure that there are no port conflicts.

  Remember when using PORT statements that the first time they appear when processing a profile data set, they completely replace the existing list. Each subsequent use of these statements within the same profile data set acts as an addition to the existing PORT list.

- DEVICE and LINK

  Provide DEVICE and LINK statements for any ATM Port for which you want SNMP ATM Management support. For example:

  ```
  DEVICE osaName ATM PORTNAME portname
  LINK linkName ATM osaName
  ```

  For a detailed explanation of the ATM DEVICE and LINK statements, see “DEVICE and LINK Statement—ATM Devices” on page 148.

Multiple TCP/IP Instances Considerations

Subagent Connection to OSA/SF

When multiple CS for OS/390 instances are active in the same MVS image, only one of the TCP/IP instances is connected to IOASNMP. In order for a TCP/IP instance to connect to IOASNMP the OSASF parameter must be specified on the SACONFIG Profile statement.

IOASNMP connects to a TCP/IP instance and acts as a server, receiving connections from those TCP/IP subagents where ATMENABLED was specified on the SACONFIG Profile statement. The result is that all these subagents connect

through the same TCP/IP to IOASNMP in order to retrieve ATM information from OSA/SF. For a depiction of this process, see Figure 40.

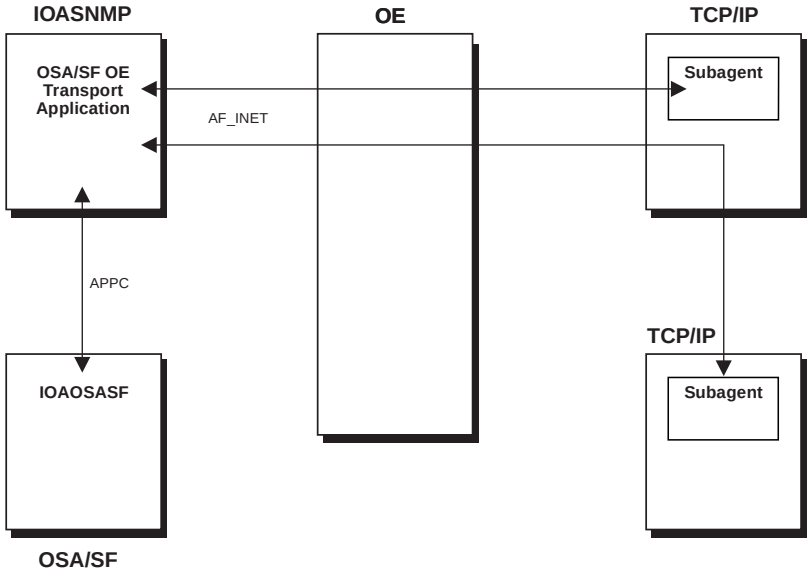


Figure 40. Subagent Connection to OSA/SF

If IOASNMP loses its connection to TCP/IP it terminates and needs to be restarted.

If the currently connected TCP/IP terminates, IOASNMP will attempt to connect to another TCP/IP instance for which the OSASF parameter was specified on the SACONFIG Profile statement, using the port number specified for the OSASF parameter. The subagents will also attempt to reconnect to OSA/SF via IOASNMP using this same OSASF port number. For this reason it is recommended that the same OSASF port number be used on the SACONFIG statement of every TCP/IP instance where the OSASF parameter is specified.

Whenever a socket error occurs on the OSA/SF socket, the connected subagents will issue the following message:

```
EZZ3219I SNMP SUBAGENT: DISCONNECTED FROM OSA/SF
```

When the subagent connection is reestablished, the following message is issued:

```
EZZ3218I SNMP SUBAGENT: CONNECTED TO OSA/SF
```

Chapter 23. Configuring the Kerberos Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure and verify the Kerberos Authentication System for TCP/IP. Also covered in this chapter are commands that can be used to control and administrate the Kerberos database.

The Kerberos system in TCP/IP consists of the following:

- Kerberos database
- Kerberos authentication server
- Remote database administration server
- Remote database administration client
- Local database administrator utilities
- Service programs
- Client programs

Configuration Process

Steps to configure Kerberos:

1. Add PORT statements to the *hlq*.PROFILE.TCPIP data set.
2. Update the authentication server cataloged procedure.
3. Update the remote database administration cataloged procedure.
4. Define the Kerberos services in *hlq*.ETC.SERVICES.
5. Create and update the Kerberos system data sets.
6. Authorize Kerberos servers.
7. Build the Kerberos database.
8. Store the Master Key.
9. Start the Kerberos servers.

A list of background information about Kerberos can be found in the bibliography. See *OS/390 SecureWay Communications Server: IP Programmer’s Reference* for more information on the application programming interface. See *OS/390 SecureWay Communications Server: IP User’s Guide* for additional commands and descriptions of the Kerberos functions.

Step 1: Add PORT Statements to the hlq.PROFILE.TCPIP Data Set

To ensure that ports TCP 750, UDP 750, TCP 751, and UDP 751 be reserved for Kerberos, add the names of the members containing the Kerberos catalogued procedures to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  750 TCP MVSKERB
  750 UDP MVSKERB
  751 TCP ADM@SERV
  751 UDP ADM@SERV
```

See “PORT Statement” on page 229 for more information about this statement.

Kerberos should not be included in the AUTOLOG list as a procedure to be started automatically when TCP/IP is invoked. Kerberos does not have a listening

connection on its reserved ports and, if placed in the AUTOLOG list, would be subject to being periodically cancelled and restarted by TCP/IP.

Step 2: Update the Authentication Server Cataloged Procedure

Update the MVSKERB cataloged procedure by copying the sample in *hlq*.SEZAINST(MVSKERB) to your system or recognized PROCLIB. Specify MVSKERB parameters and change the DD statements, as required, to suit your local configuration.

Kerberos Cataloged Procedure (MVSKERB)

```
//MVSKERB  PROC MODULE=KERBEROS,PARMS=''
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB02I
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//MVSKERB  EXEC PGM=&MODULE,
//         PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here. If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR

//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 3: Update the Remote DBA Server Cataloged Procedure

Update the remote database administration (DBA) cataloged procedure by copying the sample in *hlq*.SEZAINST(ADM@SERV) to your system or recognized PROCLIB and modifying it to suit your local configuration. Specify ADM@SERV parameters and change the DD statements, as required.

ADM@SERV Cataloged Procedure (ADM@SERV)

This is the ADM@SERV cataloged procedure:

```
//ADM@SERV PROC MODULE=ADM@SERV,PARMS=''
//*
//*        TCP/IP for MVS
```

```
//*        SMP/E Distribution Name: EZAEB02K
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//ADM@SERV EXEC PGM=&MODULE,
//           PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here. If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR

//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 4: Define the Kerberos Services in hlq.ETC.SERVICES

Before you set up the Kerberos system, the Kerberos services must be defined and their port numbers established. Add the following lines to the *hlq*.ETC.SERVICES data set:

```
kerberos             750/tcp
kerberos             750/udp
kerberos_master      751/tcp
kerberos_master      751/udp
```

The entries in *hlq*.ETC.SERVICES are case and column sensitive. They must be in lower case and begin in column 1.

Step 5: Create and Update the Kerberos System Data Sets

This step describes the data sets that you must create and update for the Kerberos system.

Kerberos Configuration Data Set (KRBCONF)

You must create either a *user_id*.KRB.CONF data set (where *user_id* is the member or proc name of the Kerberos authentication server) or an ADM@SRV.KRB.CONF data set. This data set identifies the hosts that are running the Kerberos authentication server. The *user_id*.KRB.CONF data set must be accessible by client applications.

The format of the data set is:

```
realm
realm host_name
realm host_name admin server
```

The contents of the *user_id*.KRB.CONF data set are case-sensitive.

Some examples of the data set format are:

```
univ.educ.chem
univ.other.dept joanpc
univ.educ.math chrispc admin server
```

The first line defines the local *realm* to which that host belongs. Each of the following lines specify the *realm* and *host_name* where the Kerberos server is running.

In our example, the third line lists `admin server` to indicate that the host provides an administration database server. The *host_name* that you specify must also be defined in your Domain Name Server or in the *hlq*.HOSTS.LOCAL data set.

The following sample data set can be copied from *hlq*.SEZAINST(KRBCONF).

```
raleigh.ibm.com
raleigh.ibm.com mvs2
raleigh.ibm.com mvs2 admin server
```

Note: You cannot use dotted decimal IP addresses, such as 9.67.60.11, in the KRB.CONF data set.

Kerberos Remote DBA Authorization Data Sets

To authorize the remote database administrator to add, get, and modify database entries, you must create the remote administrator authorization data sets. The data set names are:

- ADM@ACL.ADD
- ADM@ACL.GET
- ADM@ACL.MOD

You can copy these data sets from the corresponding samples:

- *hlq*.SEZAINST(ADMADD) for ADM@ACL.ADD
- *hlq*.SEZAINST(ADMGET) for ADM@ACL.GET
- *hlq*.SEZAINST(ADMMOD) for ADM@ACL.MOD

These data sets should be accessed by the remote database administration server. Your Kerberos name must be in these data sets if you want to use the KADMIN command to add, get, or modify Kerberos databases remotely.

The format of these data sets is:

```
administrator's_principal_name.admin@realm
```

realm is usually the domain name. The contents of these data sets are case-sensitive.

These data sets can contain multiple entries in the same format. Some examples of the data set format are:

```
krbadd.admin@univ.educ.chem
krbget.admin@univ.educ.bio
krbmod.admin@univ.educ.math
```

Step 6: Authorize Kerberos Servers

Using IBM's RACF program or your own security program, you need to authorize the 2 servers for access to the Kerberos database:

- The authentication server address space must have ***read*** access to the Kerberos database.
- The remote database administration server address space must have ***read/write*** access to the Kerberos database.

Step 7: Build the Kerberos Database

A special user ID named ADM@SRV must be defined in your system.

You must enter the Kerberos commands from the ADM@SRV user ID to create and use the Kerberos database before you can start the Kerberos administration server and Kerberos authentication server.

Follow these steps to create the Kerberos database:

1. Issue the KDB@INIT command to create and initialize the Kerberos database data sets. The system prompts you for the local realm.
2. At the realm prompt, enter the name of the realm where the Kerberos database resides. This is the local realm specified in the first line of the *user_id*.KRB.CONF data set. The default is YOUR_KRB.REALM. The system prompts you for the master key.
3. At the prompt, enter the master key.

   You need the master key to manage the Kerberos database. If the Kerberos database data set already exists, the system indicates that the data set already exists. Use KDB@DEST to destroy the existing database data set. See "KDB@DEST Command" on page 886 for more information.

KDB@INIT creates three database data sets:

- ADM@SRV.PRINCPL.DAT
- ADM@SRV.PRINCPL.IDX
- ADM@SRV.PRINCPL.OK

See "KDB@INIT Command" on page 889 for more information.

Step 8: Store the Master Key

The following steps describe how to store the master key.

1. On the *master_host* ADM@SRV user ID and from TSO, issue the KSTASH command.

   The system prompts you for the master key.
2. Enter `krbpass` at the password prompt.

   See "KSTASH Command" on page 891 for more information.

Step 9: Start the Kerberos Servers

This section describes how to start the following Kerberos servers:

- Authentication server
- Remote database administration server

Start the Kerberos Authentication Server

The Kerberos authentication server provides a way for authenticated users to prove their identity to other servers across a network. The authentication server reads the Kerberos database to verify that the client making the request is the client named in the request. For more information about the authentication server, which includes the ticket-granting server, see *OS/390 SecureWay Communications Server: IP Programmer's Reference*.

Before you can use the Kerberos system, you must start the Kerberos authentication server. See "Step 2: Update the Authentication Server Cataloged Procedure" on page 872 for instructions on setting up the cataloged procedure for this server.

Use the member name of the authentication server procedure as the *procname* in the START command.

The following options are available when starting the Kerberos authentication server:

-r Indicates the realm.

-l Indicates the file name. The file name must be a fully qualified data set name.

After the authentication server is started, you can use the Kerberos system. See the *OS/390 SecureWay Communications Server: IP User's Guide* for a description of the Kerberos user commands.

Maintaining the Log: When the authentication server starts, it creates a log data set called ADM@SRV.KERBEROS.LOG. All transactions to the Kerberos authentication server are recorded in this data set. Because Kerberos appends continuously, you should periodically monitor the size of the ADM@SRV.KERBEROS.LOG data set.

Stopping the Authentication Server: The authentication server can be stopped using the MVS STOP command. The authentication server will be terminated automatically when the TCP/IP address space is terminated.

Start the Kerberos Remote DBA Server

The Kerberos remote database administration server allows you to modify the Kerberos database remotely. Kerberos database data sets are accessed by the remote database administration server in write mode.

Before you can use the KADMIN command, you must start the remote database administration server. See "Step 3: Update the Remote DBA Server Cataloged Procedure" on page 872 for instructions on setting up the cataloged procedure for this server.

Use the member name of the remote database administration server procedure as the *procname* in the START command.

After the remote database administration server is started, you can use the KADMIN command on a remote host to add, retrieve, or modify the Kerberos database. See “KADMIN Command” on page 884 for more information about the KADMIN command.

Maintaining the Log: When the remote database administration server starts, it creates a log data set called ADM@SRV.ADM@SRV.LOG. All transactions to this server are recorded in this data set. Because Kerberos appends continuously, you should periodically monitor the size of the ADM@SRV.ADM@SRV.LOG data set.

Stopping the Remote Database Administration Server: The remote database authentication server can be stopped using the MVS STOP command. The remote database authentication server will be terminated automatically when the TCP/IP address space is terminated.

Verifying the Kerberos Configuration

TCP/IP for MVS provides a sample application client program and a sample application server program that can be used to verify your Kerberos installation and configuration.

This section shows an example of how to set up, run, and verify a Kerberos system.

Steps to verify Kerberos:
1. Set up the environment.
2. Register the sample service and the user.
3. Generate the key data set for the sample service.
4. Transfer the service key data set to the server.
5. Start the sample server.
6. Get the initial ticket.
7. Run the sample client program.

In this example, the hosts are assigned the following names:

Host Name	Definition
service_host	Specifies the name of the host on which the SAMPLE@S server is running.
client_host	Specifies the name of the host on which the SAMPLE@C client program is running.
master_host	Specifies the name of the host on which the Kerberos servers and the Kerberos database are running.

Note: If you run the SAMPLE@S and SAMPLE@C programs on the host where the Kerberos servers are running, you will need three IDs on the same host for verification. Verify that each host is set up correctly.

Step 1: Set Up the Environment

Follow steps 1 through 8 in “Configuration Process” on page 871.

Start the Kerberos authentication server as described in “Start the Kerberos Authentication Server” on page 876.

Note: If you fail to start the authentication server, the error messages can be seen in the output from the console.

Step 2: Register the Sample Service and the User

The following steps describe how to register the service and user.

1. On the *master_host*.ADM@SRV user ID, and from TSO, issue the KDB@EDIT command.

   The system prompts you for the Kerberos master password.
2. Enter `krbpass` at the password prompt.
3. If the master password is valid, KDB@EDIT allows you to register a service.

   The system prompts you for the *principal_name*.
4. Enter `sample` at the principal name prompt. `sample` is the service name.

   The system prompts you for the instance.
5. Enter *watson* at the instance prompt.

   The message `<not found>` is displayed and the `Create 'y'` prompt is displayed.
6. Enter a null character at the `Create 'y'` prompt to use the default value.

   The system prompts you for the password.
7. Enter `sam` at the password prompt.

   For verification purposes, the system prompts you again for the password.
8. Reenter `sam` at the password prompt for verification.
9. Enter a null character to use the default values for the remaining prompts.

   The following message is displayed if service, `sample`, is registered:

   ```
   Edit O.K.
   ```

   The system prompts you for the *principal_name*.
10. Enter `user1` at the principal name prompt.

    The system prompts you for the instance.
11. Enter a null character for a null instance at the instance prompt.

    The message `<not found>` is displayed and the `Create 'y'` prompt is displayed.
12. Enter a null character at the `Create 'y'` prompt to use the default value.

    The system prompts you for the password.
13. Enter `use` at the password prompt.

    For verification purposes, the system prompts you again for the password.
14. Reenter `use` at the new password prompt for verification.
15. Enter a null character to use the default values for the remaining prompts.

    The following message is displayed if user, `user1` is registered:

    ```
    Edit O.K.
    ```

The system prompts you for the *principal_name*.

16. Enter a null character at the principal name prompt to exit the registration process.

See “KDB@EDIT Command” on page 887 for more information about the KDB@EDIT command.

Note: *Service* refers to the host name on which SAMPLE@S is to be run.

Step 3: Generate the Key Data Set for the Sample Service

The following steps describe how to generate the key data set for the sample service.

1. On the *master_host.*ADM@SRV user ID, enter `EXT@SRVT` *watson*.

   The system prompts you for a password.
2. Enter `krbpass` at the password prompt.
3. A key data set ADM@SRV.WATSON.SRVTAB is created that contains the service keys provided by *service_host*.

   See “EXT@SRVT Command” on page 883 for more information about the EXT@SRVT command.

Step 4: Transfer the Service Key Data Set to the Server

The following steps describe how to transfer the key data set to the *server_host*.

1. Use the FTP program to transfer the ADM@SRV.WATSON.SRVTAB key data set from the *master_host* to the *service_host* as a binary transfer (EBCDIC/mode b).
2. Rename the key data set.
   - If the host providing the services is MVS, you must rename the key data set *service_id*.ETC.SRVTAB, where *service_id* is the user ID that will be starting the server.
   - If the host providing the services is VM, you must rename the key file ETC SRVTAB.
   - If the host providing the services is OS/2, DOS, UNIX, or AIX, you must rename the key file SRVTAB in the ETC directory.

Step 5: Start the Sample Server

The following describes how to start the sample server, SAMPLE@S.

1. In the server’s *hlq*.ETC.SERVICES data set, verify that the following entries exist:

   ```
   sample      906/tcp
   sample      906/udp
   ```

   where 906 is the port number that the server is using.
2. On the *service_host* or from a TSO user ID at the TSO READY prompt, invoke SAMPLE@S, to start the sample Kerberos service application.

Note: If SAMPLE@S is running in the foreground, it will run until you press the attention interupt key (on some keyboards, the attention interrupt key is ATTN). If SAMPLE@S is running in the background, it will run until you issue CANCEL.

Step 6: Get the Initial Ticket

The following steps describe how you get the initial ticket.

1. On the *client_host* or TSO user ID, issue the KINIT command to get the initial ticket.

   The system prompts you for the Kerberos name.
2. Enter `user1` at the Kerberos name prompt.

   The system prompts you for a password.
3. Enter `use` at the password prompt.

   If the KINIT command is successful, the *user_id*.TMP.TKT0 ticket data set is generated on the client host. If the KINIT command is not successful an error message is issued.

See the *OS/390 SecureWay Communications Server: IP User's Guide* for the format and description of the parameters of the KINIT command.

Step 7: Run the Sample Client Program

To start the sample Kerberos client program, enter the following on the *client_host*:

```
SAMPLE@C service_host 100
```

If all parts of your Kerberos system (environment, Kerberos database, Kerberos authentication server, and client and service applications) are set up correctly, a message is displayed as a return from the SAMPLE@S program. The following is an example of the message that might be displayed:

```
The server says:
You are user1.@UNIV.DEPT.BIO (local name user1),
at address 9.67.43.74, version VERSION X, cksum 100.
```

Setting Up a Service or Client Application

You can develop your Kerberos services and clients' applications by referencing SAMPLE@S and SAMPLE@C programs, which are provided in the *hlq*.SEZAINST data set and documented in the *OS/390 SecureWay Communications Server: IP Programmer's Reference*.

You must assign port numbers for the service in the *hlq*.ETC.SERVICES data set. We recommend that you use the port numbers that are defined in the sample *hlq*.SEZAINST(SERVICES).

Setting Up a Service Application

Use the following steps to set up a service application:

1. The database administrator uses KDB@EDIT to register the service name with the Kerberos database. The service name is used as the principal and the host name where the service is running as the instance. For example, the FTP server on the host `chrispc` would be registered as:

```
ftp.chrispc
```

You should also provide a password for the service you register. This password is converted to the key for the service.

After you register all the services provided by the same host, enter the command:

```
EXT@SRVT instance
```

For example, `EXT@SRVT chrispc` generates the ADM@SRV.CHRISPC.SRVTAB data set. The server's keys will be stored in this data set.

2. Transfer the key data set to the host providing the services (chrispc in our example).
   - You must rename the key data set *service_id*.ETC.SRVTAB, if the host providing the services is MVS.
   - You must rename the key data set ETC SRVTAB *fm*, where *fm* is the applicable file mode, if the host providing the services is VM.
   - You must rename the key data set SRVTAB in ETC directory, if the host providing the services is OS/2, DOS, UNIX, or AIX.
3. You can now start the service on the host providing the services (for example, `START chrispc`).
4. The service name must be defined in the *hlq*.ETC.SERVICES data set.

Setting Up a Client Application

The following steps describe how to set up a client application.

1. The database administration should register the client with the Kerberos database locally using KDB@EDIT or remotely using the KADMIN function.
2. Verify that the *user_id*.KRB.CONF data set at the client address space contains a valid entry. For more information see "Step 5: Create and Update the Kerberos System Data Sets" on page 873.
3. Issue the KINIT command to get an initial ticket. The ticket is saved in the *user_id*.TMP.TKT0 data set.
4. You can now start your Kerberos client application.
5. Use the KLIST command to see the ticket in the client's ticket data set. The client *user_id*.TMP.TKT0 data set contains tickets used by client applications for different servers.

   You can use the KDESTROY command to delete the ticket data set.

   See *OS/390 SecureWay Communications Server: IP User's Guide* for the format and description of the parameters of the KINIT, KLIST, and KDESTROY commands.

Administrative Commands for the Kerberos Database

Table 40 shows the Kerberos commands you can use to create and administer a Kerberos database.

Table 40. Summary of Kerberos Database Commands

Statement	Description	Page
EXT@SRVT	Generates key data sets for specified instances from ADM@SRV user ID	883

Table 40. Summary of Kerberos Database Commands (continued)

Statement	Description	Page
KADMIN	Adds, retrieves, or modifies the Kerberos database remotely from any ID	884
KDB@DEST	Erases Kerberos database data sets from ADM@SRV user ID	886
KDB@EDIT	Registers users to the Kerberos database from ADM@SRV user ID	887
KDB@INIT	Builds and formats the Kerberos database from ADM@SRV user ID	889
KDB@UTIL	Loads or dumps the Kerberos database from ADM@SRV user ID	890
KSTASH	Stores the Master Key	891

EXT@SRVT Command

Use EXT@SRVT command to generate a key data set for instances.

A key data set contains all the service keys associated with the servers running with the same instance. An instance is usually the host name where the services are provided. The service keys in the key data sets are used by the servers to verify whether a ticket presented by a user is legal.

You should rename this data set and send it to a server. It contains a copy of the service keys and is used for authentication of a client's request by the remote server.

Syntax

Parameters

instance
The host name for which a report is to be generated.

Examples

The system searches through the Kerberos database entries for each of the specified instances. For example, if you enter EXT@SRVT `INST1`, the system searches through the Kerberos database entries for the specified instance of `INST1`. When the system finds `INST1`, a key data set ADM@SRV.INST1.SRVTAB is generated.

```
EXT@SRVT INST1 INST2
Enter Kerberos master key.  <krb_pw>

Current Kerberos master key version is

Master Key entered.  BEWARE!

generating INST1 SRVTAB
generating INST2 SRVTAB
```

Usage Notes

- Multiple instances can be specified on the same command line to generate multiple key data sets.
- You can use the KLIST -srvtab command to see the contents of a key data set. See *OS/390 SecureWay Communications Server: IP User's Guide* for the usage of the KLIST command.
- Copy each instance SRVTAB key data set to the corresponding instance's host, and rename the key data set to *user_id*.ETC.SRVTAB.

KADMIN Command

Use the KADMIN command to add, get, or modify a Kerberos database.

You can only use this command for a Kerberos user with instance as `null`. In addition, as a remote administrator with instance as `admin`, you can change your own password.

Syntax

Parameters

None.

Examples

After issuing the KADMIN command, you will be prompted for your user ID. When you enter the administrator's *principal_name*, the following message is displayed:

```
Welcome to the Kerberos Administration Program, Version X
Type "help" if you need it.
admin:
```

At the `admin:` prompt, you can enter `?` to list the available subcommands.

Usage Notes

- In order to use the KADMIN command you must:
  1. Have the remote administration server, ADM@SERV, running on the machine that contains the Kerberos database when you issue the command.
  2. Register the remote Kerberos administrator, although no special user ID name is required for running KADMIN. You can register with the Kerberos database using KDB@EDIT from the ADM@SRV user ID. Make the Kerberos instance be `admin`.
  3. Have your Kerberos name in the remote administrator authorization data sets to perform the corresponding database operations.
- You can use the following subcommands once you are in the Kerberos Administration program:

  ADD_NEW_KEY
  Registers a new *principal_name* with the Kerberos database. Requires one argument, the *principal_name*. (The ADD_NEW_KEY subcommand can be abbreviated to ANK.)

  CHANGE_ADMIN_PASSWORD
  Changes your ADMIN *instance* password. Requires no arguments. (The CHANGE_ADMIN_PASSWORD subcommand can be abbreviated to CAP.)

  CHANGE_PASSWORD
  Changes the Kerberos password for the *principal_name*. Requires one argument, the *principal_name*. (The CHANGE_PASSWORD subcommand can be abbreviated to CPW.)

EXIT
Ends the Kerberos program. Alternatively, you can use the QUIT subcommand.

GET_ENTRY
Gets you an entry into the Kerberos database for review. At the password prompt, enter your administrator password. (The GET_ENTRY subcommand can be abbreviated to GET.)

HELP *command name*
Displays help messages for KADMIN. If you enter this subcommand without an argument, a general help message is displayed.

LIST_REQUESTS
Displays a list of possible subcommands. (The LIST_REQUESTS subcommand can be abbreviated to LR or ?.)

QUIT
Ends the Kerberos program. Alternatively, you can use the EXIT subcommand.

Related Topics

- “Kerberos Remote DBA Authorization Data Sets” on page 874
- “KDB@EDIT Command” on page 887

KDB@DEST Command

Use the KDB@DEST command to erase the ADM@SRV.PRINCPL.DAT and ADM@SRV.PRINCPL.IDX data sets.

Syntax

Parameters

None.

KDB@EDIT Command

Use the KDB@EDIT command to register users and services to the Kerberos database.

The system prompts you for the Kerberos master key and for information about the user and service that you want to add.

Syntax

Parameters

None.

Examples

This example shows how to register a Kerberos user or service using the KDB@EDIT command. After issuing the KDB@EDIT command:

1. When the system prompts you for the Kerberos master key, enter the valid master key at the master key prompt:

```
Opening database...

Enter Kerberos master:password:<krb_pw>
```

   The system prompts you for the principal name.

2. Enter the user name or service name at the principal name prompt:

```
Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in 'brackets'
enter return to leave the same, or new value.

Principal name: <username>
```

3. The system prompts you for the instance:

```
Instance: <enter>
```

   - If you are registering a user, enter a null instance.
   - If you are registering a remote administrator, enter `admin`.
   - If you are registering a service, at the instance prompt, enter the name of the host where the service resides.

   The system prompts for more information, depending on whether the user or the service already exists in the database.

   If the user or the service already exists in the database, the system asks you if you want to change the password.

```
<Not found>, Create 'y' ? <y>
Principal: username, Instance: , kdc_key_ver: 1
password: <userpassword>

Verifying, please reenter
New Password: <userpassword>
```

If the user or the service does not exist in the database, the system prompts you for the expiration date, ticket lifetime for the user or services, and the attribute.

```
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) '1999-12-31' ? <enter>
Max ticket lifetime (*5 minutes) '255 - ? <enter>
Attributes '0 - ? <enter>
```

Defaults are provided for these prompts. If you want to use the default values, enter a null character. Otherwise, enter the desired value.

There are two ways to enter a null character, depending on your system.

- Press the **SPACE** bar and then the **ENTER** key
- Press the **ENTER** key.

The following message is displayed if the user or service is registered:

```
Edit O.K.
Principle name:  <username>
```

The system prompts you for the next entry.

4. Enter a null character at the principal name prompt to exit the registration process or repeat the process to register the next Kerberos user name or Kerberos service name that you want to establish.

You must inform the user of the Kerberos name and password that you assign to that user.

Usage Notes

An instance is usually the host name where the services are provided. A user's instance and a service's instance are usually specified by the following rules:

- A user's instance is optional. Users with privileges (for example, the remote system administrator) should register with an instance of `admin`. Users without privileges have an instance of `null`.
- A service's instance is usually the host name where the service is running.

Related Topics

See "KDB@UTIL Command" on page 890.

KDB@INIT Command

Use the KDB@INIT command to create and format the Kerberos database.

Syntax

Parameters

None.

Examples

After issuing the KDB@INIT command, the system will prompt you for the local realm. At the realm prompt, enter the name of the realm where the Kerberos database resides. This is the local realm specified in the first line of the *user_id*.KRB.CONF data set. The default is YOUR_KRB.REALM.

Then the system will prompt you for the master key. At the prompt, enter the master key.

Usage Notes

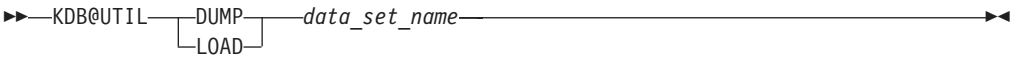
KDB@INIT creates 3 database data sets:

- ADM@SRV.PRINCPL.DAT
- ADM@SRV.PRINCPL.IDX
- ADM@SRV.PRINCPL.OK

KDB@UTIL Command

Use the KDB@UTIL to dump or load the Kerberos database.

Syntax



Parameters

DUMP
Dump the contents of the Kerberos database to the specified *data_set_name*.

LOAD
Load the contents of the specified *data_set_name* to the Kerberos database.

data_set_name
Specifies the data set name of the text data set into which the database is dumped or from which the database is loaded.

Examples

The following example uses the KDB@UTIL command to examine the contents of the database. First, the database was dumped to a data set named ABC.DB with the command:

```
KDB@UTIL DUMP ABC.DB
```

Then, the ABC.DB was edited using a system editor.

```
K M 255 1 1 0 6B81 2e63 200001010459 199001082204 db_creation *
changepw KERBEROS 255 1 1 0 71d0 a1f7 200001010459 199001082204 db_creation *
default *255 1 1 0 0 0 200001010459 199001082204 db_creation *
krbtgt YOUR_KRB.RE.ALM 255 1 1 0 65f8 7291 200001010459
199001082204 db_creation *
```

Note: Each line is a database entry.
Kerberos names were deleted and then the database was reloaded using the following command:

```
KDB@UTIL LOAD ABC.DB
```

Usage Notes

You can edit the dumped data set using a system editor to add, change, or delete the contents of the database.

KSTASH Command

Use the KSTASH command to create the ADM@SRV.ETC.K data set. This data set is required for MVSKERB and ADM@SERV tasks.

Syntax

Parameters

None.

Usage Notes

You will be prompted for the Kerberos master key. The master key is case-sensitive.

Chapter 24. Configuring the Remote Print Server (LPD)

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

The Remote Print (LPD) server supports the Line Print Daemon and allows you to print on JES controlled printers from any host in your TCP/IP network that implements the Line Print client functions. These client functions are invoked with the LPR command.

Refer to “Starting and Stopping TCP/IP Servers” on page 261 for information on starting and stopping the TCP/IP print server (LPD). When LPD is stopped by the MVS operator with the *P procname* command, LPD will terminate any TCP/IP connections currently transferring data. Before ending, LPD will finish spooling to JES any print jobs that it has received and is currently spooling. JES will handle these jobs after LPD ends.

This chapter describes how to configure the LPD server and use the SMSG interface to operate it once it is running.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Configuration Process

Steps to configure the Remote Print Server:

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the LPD server catalogued procedure.
3. Update the LPD server configuration data set.
4. Create a banner page (optional).

For information about operating and controlling the LPD server, see “Operating the LPD Server Using the SMSG Interface” on page 916.

Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP

If you want the LPD server started automatically when the TCPIP address space is started, include the name of the member containing the LPD server cataloged procedure in the AUTOLOG list in *hlq*.PROFILE.TCPIP. For example:

```
AUTOLOG
  LPSERVE
ENDAUTOLOG
```

To ensure that port TCP 515 is reserved for the LPD server, also add the name of the member containing the LPD cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP. For example:

```
PORT
  515 TCP LPSERVE
```

See “AUTOLOG Statement” on page 137 for more information about the AUTOLOG statement. See “PORT Statement” on page 229 for more information about the PORT statement.

Step 2: Update the LPD Server Cataloged Procedure

Update the LPD server cataloged procedure to suit your local configuration by copying the sample to your system or recognized PROCLIB from *hlq*.SEZAINST(LPSPROC) and modifying the sample. Specify LPD parameters, and change the data set names as required. See “Specifying LPD Server Parameters” on page 895 for more information on the LPD server parameters.

LPD Server Cataloged Procedure (LPSPROC)

```
//LPSERVE  PROC MODULE=LPD,
//         LPDDATA=TCPIP.SEZAINST(LPDDATA),
//         LPDPRFX='PREFIX TCPIP',
//         DIAG=''
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB019
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//*        Turn on SMSG support
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//LPD   EXEC PGM=MVPMAIN,
//  PARM=('&MODULE,ERRFILE(SYSERR),HEAP(512)',
//       'NOSPIE/ ''&LPDDATA'' &LPDPRFX &DIAG'),
//       REGION=6M,TIME=1440
//SPOOL OUTPUT CHARS=GT12
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//LPD1 OUTPUT CHARS=GT12
//*
//*        SYSPRINT contains runtime diagnostics from LPD. It
//*        can be a data set or SYSOUT.
//*
//SYSPRINT DD SYSOUT=*
//*
//*        SYSERR contains runtime diagnostics from Pascal. It can be
//*        a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*        SYSDEBUG receives output that is generated when the TRACE
//*        parameter is specified in the PARM on the EXEC card.
//*        It can be a data set or SYSOUT.
//*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
```

```
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
```

Specifying LPD Server Parameters

The system parameters required by the LPD server are passed by the PARM option on the EXEC statement of the LPD cataloged procedure. Update the following parameters as required.

LPDDATA='*data_set_name***'**
Specifies the fully qualified name of the data set containing the LPD configuration statements. This data set can be sequential or a member of a PDS.

LPDPRFX='PREFIX *your_prefix***'**
Specifies the high-level qualifier to be used for temporary data sets created by the LPD server. Include both the PREFIX keyword and your qualifier in the quoted string. The qualifier may be up to 26 characters. If it is blank, it defaults to the procedure name. The LPD task requires the authority to create and modify data sets with this prefix.

DIAG='*options***'**
Specifies any of the following diagnostic options in a quoted string of keywords separated by blanks. For example, `DIAG='VERSION TRACE'`

VERSION
Displays the version number.

TYPE Activates high-level trace facility in the LPD server. Significant events, such as the receipt of a job for printing, are recorded in the //SYSOUT DD data set specified in your LPD server cataloged procedure.

TRACE
Causes a detailed trace of activities within the LPD server to record in the //SYSOUT DD data set specified in your LPD server cataloged procedure. The detailed tracing can be also activated with the DEBUG statement in the LPD server configuration data set and with the TRACE command of the SMSG interface.

Note: The JCL PARM= statement has a limit of 100 characters.

Alternative Method for Specifying the LPD Configuration File

Use a DD card that requires the LPD configuration file to be in a data set.

```
//SYSLPDD  DD DISP=SHR,DSN=TCPIP.V2R7.SEZAINST(LPDDATA)
```

Using this example, the DD card must be specified as SYSLPDD, but the data set name can be any valid data set name with a member specified up to 44 characters.

In order to use the DD card method, you **must** comment out the LPDDATA= and remove ″&LPDDATA″; from the PARM= statement.

Note: The search order for the configuration file is:
1. LPDDATA= on the PARM= statement
2. //SYSLPDD DD statement
3. *hlq*.LPD.CONFIG

If both the LPDDATA= statement on the PARM= statement and the //SYSLPDD DD statement are specified, the data set name specified on LPDDATA= is used.

The LPD server does not limit the number of print jobs it handles per connection. This can cause a memory abend to occur if too many print jobs are sent in one connection. Certain LPR clients, such as SUN UNIX, are set up to send multiple jobs in one connection. It is recommended that the LPD start procedure be started with a region size of 6M and the LPR client send no more than 50 print jobs in one connection.

Step 3: Update the LPD Server Configuration Data Set

The LPD configuration data set defines the local, NJE, and remote services (printers and punches) used by the LPD server. To update the LPD server configuration data set, copy the sample provided in *hlq*.SEZAINST(LPDDATA) and modify it to suit your installation.

- To define a printer or punch:
  - Include a SERVICE statement with the appropriate parameters for each printer or punch your are defining.
  - Specify the type of service with the LOCAL, NJE, or REMOTE parameter in the SERVICE statement.
  - For local or NJE services, include any of the optional parameters to further define the service: EXIT, FAILEDJOB, FILTERS, LINESIZE, PAGESIZE, RACF, and SMTP.
  - For remote services, specify the destination printer and host. Any additional specifications are defined on the remote system and are not necessary in the SERVICE statement.
- To turn on LPD server tracing, include the optional DEBUG statement.
- To authorize users for the SMSG interface, include the optional OBEY statement.
- Printer names cannot contain an at sign (@).

Summary of LPD Server Configuration Statements

The valid statements for this data set are listed in the following table.

Table 41. Summary of LPD Server Configuration Statements

Statement	Description	Page
DEBUG	Turn on tracing of all LPD processes.	900
JOBPACING	Restricts parallel processing of jobs to conserve memory.	901
OBEY	Specify users IDs that can use the SMSG interface.	902
SERVICE	Specify the name and type of service.	903
STEPLIMIT	Restricts complexity of jobs received to conserve memory.	913

Table 41. Summary of LPD Server Configuration Statements (continued)

Statement	Description	Page
UNIT	Specifies type of DASD that LPD should use for temporary data sets.	914
VOLUME	Specifies the volume that LPD should use for temporary data sets.	915

Sample LPD Server Configuration Data Set (LPDDATA)

```
;LPD CONFIGURATION DATA SET
;==========================
;
; COPYRIGHT = NONE.
; Change Activity
; $L1=MV11199 HTP320 951206 RTPMCL: Added comments
; $01=PN66730 HTP310 960131 RTPMCL: Added change flag
;
; This data set describes the printers and punches (which are both calle
; SERVICE) that are usable from LPR client programs for this host.
;
; Each SERVICE must be described as LOCAL, NJE, or REMOTE.  Data for
; LOCAL services are managed directly by JES.  Data for NJE services
; are managed by NJE.  REMOTE services' data are forwarded to another
; LPD (print server).
;
; You can control which types of printing or punching can be done
; through a particular SERVICE with FILTERS.  The 4 currently
; available FILTERS are:
;
;       f    which paginates the data set at the size of the page given.
;            It also truncates lines if they exceed a maximum length.
;       l    which does not insert pagination but will truncate lines
;            as the "f" filter does.
;       p    which paginates the data set, adding titles, the date, and
;            page numbers as well as providing line truncation.
;       r    which prints the data set, interpreting the first column
;            of each line as FORTRAN carriage control.
;
;
; Most printer SERVICEs should allow all three but you probably only
; want to specify "l" for punches.
;
; The LINESIZE option can be used to limit the length of lines written
; by the filters.
;
; The PAGESIZE option can be used for filters that do pagination to
; specify how many lines should appear on a page.
;
; The RACF option will cause the server to verify that a user knows
; the account password for a user ID on this host.
;
; These statements define a LOCAL PRINTER SERVICE called locprt1, which
; is a conventional printer that will use the JES printing facilities.
;
;DEBUG
 SERVICE locprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
;
; These statements define an NJE PRINTER SERVICE called njeprt1, which
; provides access to the NJE service on this system.
;
;SERVICE njeprt1 PRINTER
```

```
;   NJE DEST=RALVMM IDENTIFIER=JOHN OUTPUT=LPD1
;   FILTERS f l p r
;   LINESIZE 132
;   PAGESIZE 60
;
; These statements define a REMOTE SERVICE called pebprt, which
; provides access to the printing queues on another system.
; From an LPR client, specify the printer name defined on the SERVICE
; statement and the hostname or IP address of the host that this LPD
; is running on, NOT the names on the REMOTE statement.
;     Example: LPR fn (p pebprt h LPDSrvHostName
; The above is required if you wish to send the data to the REMOTE
; printer via this LPD.
;
;SERVICE pebprt PRINTER
;  REMOTE lpt1@PEBBLES.TCP.RALEIGH.IBM.COM
; FAILEDJOB MAIL
;
; These statements define a PUNCH SERVICE called pun1, which
; provides access to the JES controlled PUNCH.
;
;SERVICE pun1 PUNCH
;   LOCAL
;   FILTERS l
;   LINESIZE 80
;
```

Step 4: Create a Banner Page (Optional)

LPBANNER is the name of the default program that is provided in executable form in the SEZATCP data set. This program is specified on the EXIT parameter in the SERVICE statement. LPBANNER prints a separator page that contains, in large letters, a banner stating “LPD BANNER”, the user ID, the jobname, and the job class. Field headings of HOST, USER, JOB, and CLASS appear in smaller letters.

The sample exit LPBANNER uses machine carriage control and is designed to be used with the SERVICE PRINTER LOCAL or SERVICE PRINTER NJE statements. Banner pages are usually not used with the SERVICE PUNCH or SERVICE RECFMU statements; however, if you want to have banner pages (headers) for the SERVICE PUNCH or SERVICE RECFMU devices, a user created banner exit is required.

You can either use the executable form or copy and modify the sample source provided in *hlq*.SEZAINST(EZAAE04S) and *hlq*.SEZAINST(EZAAE04T) to create a banner. If you are changing the source to create your own banner, assemble and link-edit these data sets as re-entrant. You can modify and use the following JCL to do this. Changing the ENTRY and NAME to something other than LPBANNER will avoid possible maintenance problems in the future.

```
//ASMLNK JOB MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A,REGION=1024K
//ASM1   EXEC PGM=ASMA90,PARM='OBJECT,XREF(FULL)'
//STEPLIB DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip..V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04S),DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(5,5,1)),DCB=BLKSIZE=400
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04S),DISP=SHR
/*
```

```
//ASM2   EXEC PGM=ASMA90,PARM='OBJECT,NODECK,XREF'
//STEPLIB  DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip.V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04T),DISP=(OLD,PASS)
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04T),DISP=SHR
/*
//LNK    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,LET'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD DSN=tcpip.V3R4.SEZATCP,DISP=SHR
//AEZAMOD1 DD DSN=tcpip.V3R4.AEZAMOD1,DISP=SHR
//OBJECT   DD DSN=&&OBJECT,DISP=(OLD,DELETE)
//SYSLIN   DD *
  ORDER   EZBOECPR
  INCLUDE AEZAMOD1(EZBOECPR)
  INCLUDE OBJECT(EZAAE04S)
  INCLUDE OBJECT(EZAAE04T)
  MODE    AMODE(24),RMODE(24)
  ENTRY   LPBANNER
  NAME    LPBANNER(R)
/*
```

Statements for the LPD Server Configuration Data Set

This section includes the syntax rules and alphabetically listed definitions of the statements for the data set used to configure the LPD Server,

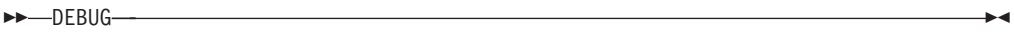
Syntax Rules

In the LPD server configuration data set, tokens are delimited by blanks and record boundaries. All characters to the right of and including a semicolon are treated as comments.

DEBUG Statement

Use the DEBUG statement to activate full tracing of the processing within the LPD server.

Syntax



Parameters

None.

Usage Notes

Detailed tracing can also be activated using the TRACE parameter on the PROC statement of the LPD server procedure or by specifying TRACE ON with the SMSG interface. The DEBUG statement can be placed anywhere in the data set but will only affect those services following it. Including DEBUG as the first statement in the configuration data set allows trace messages to be written from the point LPD is initialized.

Related Topics

- “Specifying LPD Server Parameters” on page 895
- “Operating the LPD Server Using the SMSG Interface” on page 916

JOBPACING Statement

Use the JOBPACING statement to limit the number of jobs that the LPD server will concurrently write to the JES spool or send to another LPD server. This limits memory requirements in LPD, but does not cause any jobs to be lost. Received jobs are queued until they can be processed.

Syntax

Parameters

limit

An integer specifying the maximum number of jobs that the LPD server will concurrently write to the JES spool or send to another LPD server.

Usage Notes

- Concurrent processing of jobs requires memory for control blocks and large I/O buffers. Some concurrent job processing keeps a long job or slow receiving LPD from delaying all the other jobs. Too much concurrent processing causes thrashing and requires lots of memory.
- JOBPACING defaults to the recommended value of 5 when the keyword is not specified. Increasing this value may cause memory allocation problems with certain system configurations.
- If LPD runs out of memory, reduce the value of either JOBPACING or STEPLIMIT.

OBEY Statement

Use the OBEY statement to specify user IDs authorized to use the SMSG interface provided with LPD server.

Syntax

Parameters

user_id
The user IDs authorized to use the SMSG interface. See “Operating the LPD Server Using the SMSG Interface” on page 916 for more information.

Examples

The following statement allows three test user IDs to use the SMSG interface:

```
OBEY TESTER01 TESTER02 TESTER03
```

Usage Notes

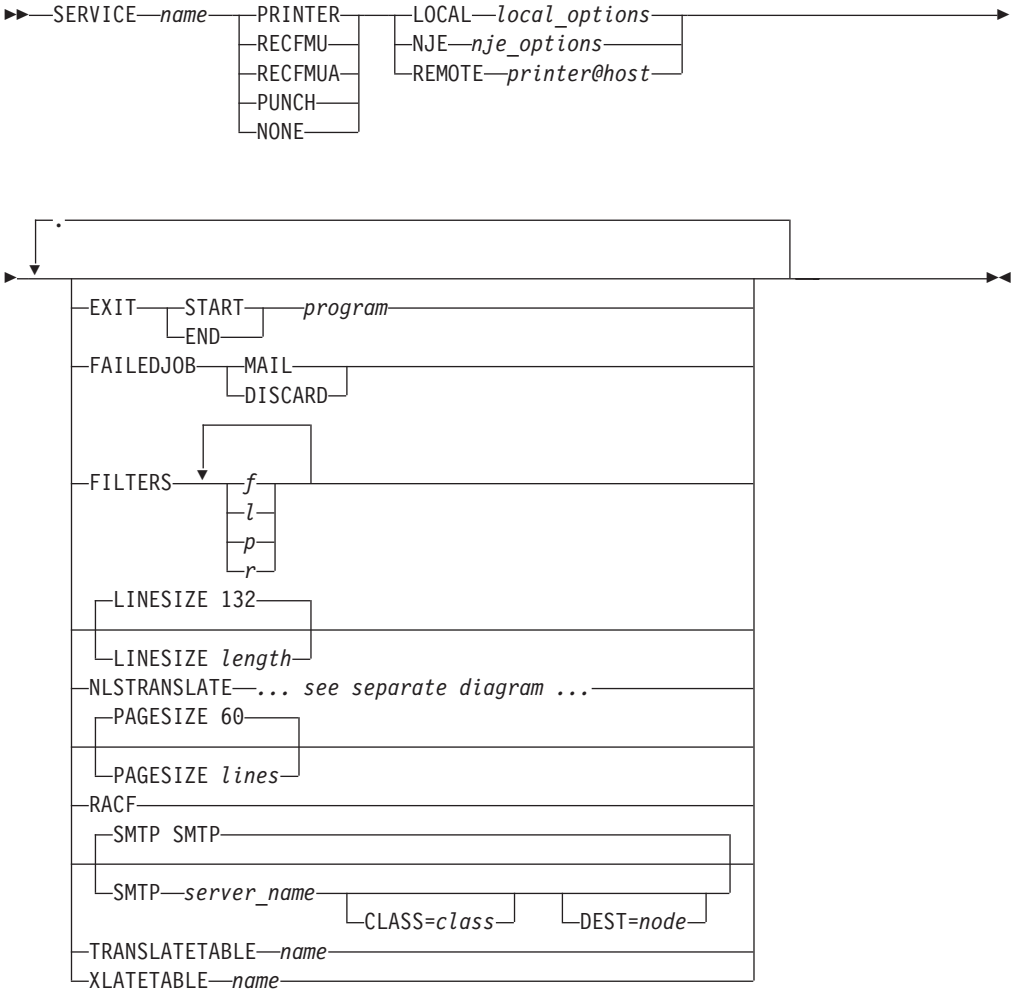
Multiple user IDs can be specified on the OBEY statement. More than one OBEY statement can be included in the data set.

SERVICE Statement

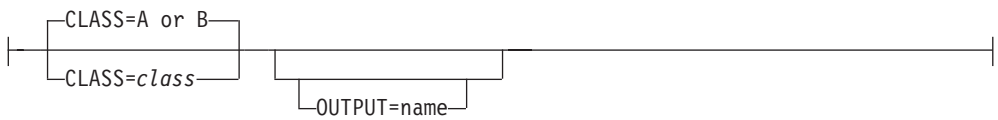
Use the SERVICE statement to specify the name and type of service for the printers and punches used by the LPD server. This service name is used in the LPR command.

Note: The parameters shown on separate lines must be coded on separate lines. Follow the example in the sample configuration data set shown in “Sample LPD Server Configuration Data Set (LPDDATA)” on page 897.

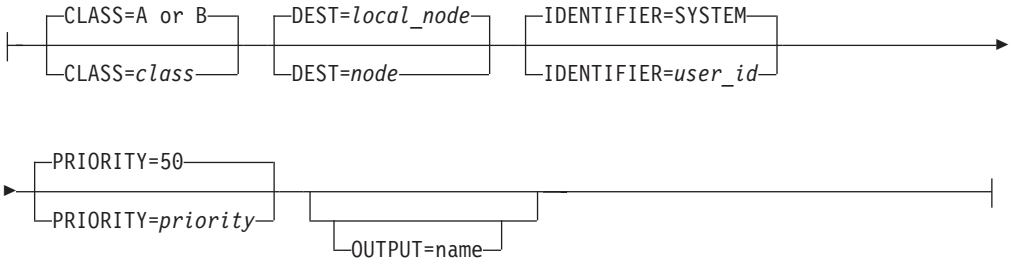
Syntax



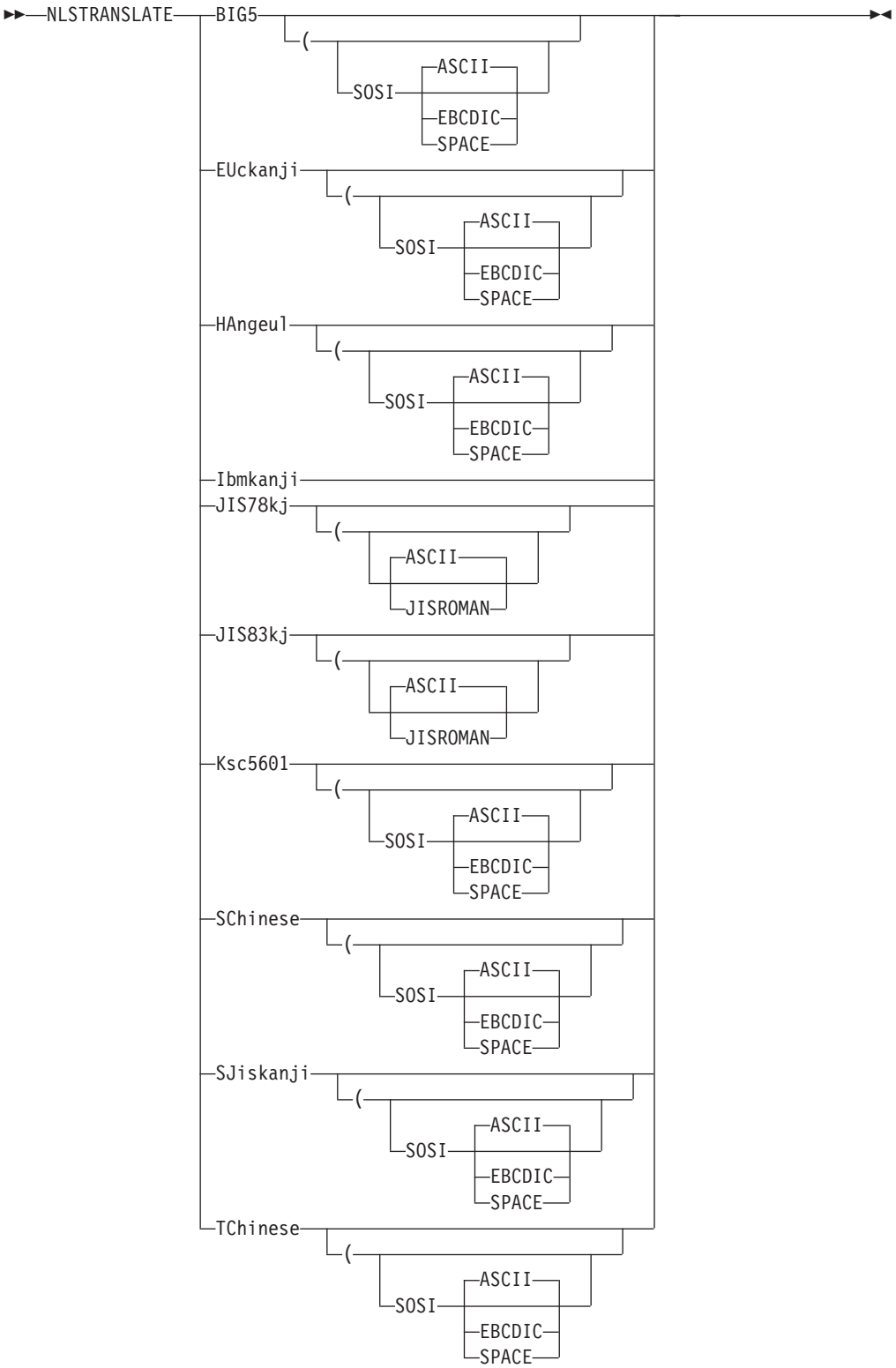
local_options:



nje_options:



Syntax



Parameters

name

The service name must be 1–8 characters in length. Only characters permitted in MVS data set names are valid. This value is case sensitive.

PRINTER
Specifies that the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UM and machine carriage control is written in column 1 of the file. For filter ″1″, this will always be single space (’09’X). For other filters, it will be determined from the data received.

RECFMU
Specifies that the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=U and carriage control characters are not added to the beginning of each line. The JES spool file is like PRINTER, but carriage control is not added.

RECFMUA
Specifies that the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UA. The carriage control (CC) character is taken from the first column of user data after any LPD processing. Only filter ″1″ should be allowed with this device type. Specify ″filters 1″ in the SERVICE statement so LPD will not print jobs requesting other filter options. The LPD trace would show message EZA0801I for these aborted jobs.

PUNCH
Specifies that the service is to a punch device. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UM and machine carriage control is written in column 1 of the file.

NONE
Specifies that the service is not currently in use.

LOCAL
Specifies that the data sets sent to a service are written to the local MVS printer or punch.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

OUTPUT=*name*
Specifies the name of an OUTPUT DD statement that contains additional spool parameters.

NJE
Specifies that the data sets sent to a service are delivered to the NJE system.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

DEST=*node*
The name of the NJE node. The default is the local node.

IDENTIFIER=*user_id*
The device user ID. The default is SYSTEM.

PRIORITY=*priority*
Specifies the transmission priority. The default is 50.

OUTPUT=*name*
Specifies the name of an OUTPUT DD statement that contains additional spool parameters.

REMOTE
Specifies that data sets sent to a service are forwarded to a specified remote printer.

printer@host
The destination printer at a specified internet host. This can be an internet name or an IP address.

EXIT
Specifies any program to be executed before closing, but after allocating and opening, an output data set.

START
Specifies that the program is invoked after allocating and opening the output data set, but before anything is written to the data set. This parameter is mutually exclusive of the END parameter.

END
Specifies that the program is invoked just before closing the output data set. This parameter is mutually exclusive of the START parameter.

program
Name of the program to be invoked. See “Step 4: Create a Banner Page (Optional)” on page 898 for information on using the default LPBANNER or creating your own banner page program. The library containing the program should be in the system’s link list, (LNKLSTxx) or a STEPLIB definition can be used if the library is APF authorized.

FAILEDJOB
Specifies whether a notice of failed jobs should be mailed to users or a job is discarded without notice.

MAIL
Specifies that notices of failed jobs are mailed to users.

Note: To use the MAIL parameter, you must also specify the SMTP parameter. Messages are logged in the LPD joblog, showing the information sent to SMTP.

DISCARD
Specifies that failed jobs are discarded without notice.

FILTERS
The ″filter″ specified in the control file received by LPD specifies how LPD should format the job. LINESIZE and PAGESIZE specified on the SERVICE statement can also affect the formatting.

The FILTERS parameter on the SERVICE statement controls the types of printing or punching allowed by the line printer daemon (LPD). Most printer services should allow all of the FILTERS parameters. However, you may want to specify only ″l″ for punches or RECFMUA type printers. A job that requests a filter that is NOT allowed will be abandoned. It will not print, and LPD will issue an EZB0801I message.

The control file received by LPD specifies the filter actually used. LPD formatting for each possible filter is described below:

f **- Print formatted file**
Paginates the data set at the size of the page given. It also truncates lines if they exceed a maximum length.

This filter causes the data file to be printed as a plain text file, providing page breaks as necessary. Only ASCII control characters in the following list are honored: HT, CR, FF, LF, VT, and BS. They are removed from the

datastream (not printed) and changed into equivalent spacing and machine carriage control. Any ASCII code that translates to an EBCDIC NL is also honored, however, standard ASCII tables do not have a NL (new line) control character.

JES writers start each job on a new page. Therefore, LPD suppresses any FF (form feed) at the beginning of the data to avoid an extra page eject before the user's data set is printed.

/ **- Print file leaving control characters**
Does not insert pagination but does truncate lines. All lines are single spaced.

This filter causes the specified data file to be printed without filtering out control characters (except LF which is used to determine line endings when converting to a JES record oriented spool file). Other ASCII control characters will be translated to EBCDIC and printed as text. They will NOT be converted to equivalent machine carriage control. Use filter ″f″ when you want control codes like FF and HT to be honored.

p **- Print file with 'pr' format**
Paginates data set, adding titles, the date, and page numbers as well as providing line truncation.

This filter causes the data file to be printed with a heading, page numbers, and pagination. Page breaks are determined by the PAGESIZE configuration on the SERVICE statement, or by ASCII FF (form feed) control characters in the datastream. PAGESIZE includes the title lines printed.

JES writers start each job on a new page. Therefore, LPD suppresses any FF (form feed) at the beginning of the data to avoid an extra page eject before the user's data set is printed.

r **- File to print with FORTRAN carriage control**
Prints the data set, interpreting the first column of each line as a FORTRAN carriage control. The FORTRAN contols are removed from the datastream and translated into equivalent machine carriage control. LPD honors ″ ″, ″1″, ″0″, ″+″, and ″-″. Other values in column 1 cause single spacing. LPD also truncates lines if they exceed LINESIZE. Page breaks are determined by the PAGESIZE configuration as well as the Fortran controls in column 1.

LINESIZE
Specifies the line length used by the filters when they truncate lines. This statement only applies to services that are designated as either LOCAL or NJE. on PAGESIZE (for example, 100000).

length
The number of characters in a line on a page. Lines longer than this number are truncated. The default is 132.

PAGESIZE
Specifies the page length used by the filters when they paginate.

This statement only applies to services that are designated as either LOCAL or NJE.

lines
The number of lines on a page. The default is 60.

RACF
Allows control of which users print data sets on this service.

SMTP
Specifies the SMTP server name, CLASS, and DEST options for sending failed jobs notices. (For additional information, see the description of the FAILEDJOB MAIL parameter.

server_name
Specifies the name of the SMTP server. If this statement is omitted, the default is SMTP.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

DEST=*node*
The NJE node to which SMTP messages should be sent.

TRANSLATETABLE
Specifies the translation table in the *name*.TCPXLBIN data set to be used by the client. XLATETABLE is a synonym for this parameter.

name
Specifies the name of the SBCS translation table.

XLATETABLE
Specifies the translation table in the *name*.TCPXLBIN data set to be used by the client. TRANSLATETABLE is a synonym for this parameter.

name
Specifies the name of the SBCS translation table.

NLSTRANSLATE
Specifies the DBCS translation type to be used when a client selects the named SERVICE.

BIG5
Select the translation type from Big-5 PC DBCS codes to Traditional Chinese host codes.

EUckanji
Select the translation type from Japanese EUC DBCS codes to Japanese host codes.

HAngeul
Select the translation type from Korean PC DBCS codes to Korean host codes.

Ibmkanji
This option actually causes no conversion to be performed; in other words, data is sent to a printer without translation. Ibmkanji may be used for sending data in EBCDIC. If you select this option, be sure other printers on the same network are all configured with Ibmkanji.

JIS78kj
Select the translation type from JIS 1978 DBCS codes to Japanese host codes. The Escape Sequence, `ESC 2/4 4/0`, is used to express JIS X0208 1978.

JIS83kj
Select the translation type from JIS 1983 DBCS codes to Japanese host codes. The Escape Sequence, `ESC 2/4 4/2`, is used to express JIS X0208 1983.

Ksc5601
Select the translation type from IBM KS DBCS codes to Korean host codes.

SChinese
Select the translation type from Simplified Chinese PC DBCS codes to Simplified Chinese host codes.

SJiskanji
Select the translation type from Shift JIS DBCS codes to Japanese host codes.

TChinese
Select the translation type from Traditional Chinese 5550 PC DBCS codes to Traditional Chinese host codes.

SOSI
Shift-Out and Shift-In characters X'1E' and X'1F' are used in data to delimit DBCS strings.

SOSI ASCII
Shift-Out and Shift-In characters X'1E' and X'1F' are used in data to delimit DBCS strings.

SOSI EBCDIC
Shift-Out and Shift-In characters X'0E' and X'0F' are used in data to delimit DBCS strings.

SOSI SPACE
Shift-Out and Shift-In characters X'20' and X'20' are used in data to delimit DBCS strings.

ASCII (with JIS78KJ and JIS83KJ only)
The ASCII Escape Sequence, `ESC 2/8 4/2`, is used in data to express SBCS strings.

JISROMAN (with JIS78KJ and JIS83KJ only)
The JISROMAN Escape Sequence, `ESC 2/8 4/10`, is used in data to express SBCS strings.

Examples

- PRINTER and PUNCH definitions

  The sample configuration data set SEZAINST(LPDDATA) provides examples of SERVICE statements for LOCAL, REMOTE, and NJE printers and a LOCAL punch.

- EXIT Parameter

  To make the LPBANNER program print a page at the beginning of the printed output, use the EXIT START parameter within a SERVICE statement.

```
SERVICE locprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
    EXIT START LPBANNER
```

  To make the LPBANNER program print a page at the end of the printed output, use the EXIT END parameter within a SERVICE statement.

```
SERVICE locprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
    EXIT END LPBANNER
```

Refer to RFC1179 *Section 7.5 Line Printer Daemon Protocol* for more information about the LPD user exit.

Usage Notes

- For remote printers:

  Remote printers do not require specifications for EXIT, FAILEDJOB, FILTERS, LINESIZE, PAGESIZE, RACF, SMTP, and translation tables. These are defined on the remote system.

  The LPR command must be specified with the printer name as it is specified on the SERVICE statement. The HOST parameter may be HOSTNAME or the IP address of the host the LPD is running on – **not** the printer name and IP address of the remote printer.

  ```
  LPR fn (p pebprt h LPDSrvHostName
  ```

  This is required if you want to send data to the remote printer via this LPD.

- With RACF:

  In order to print data sets on a printer that has RACF specified, the user must use the JOB option with a valid password on the LPR command.

  If the RACF keyword is specified for the service and a valid password is not supplied, the job sent to that service will fail.

  If a printer is defined as RACF for a local service on one system and as an NJE service on other systems, then you must specify the RACF keyword on the SERVICE statement on each of the systems where this service is defined.

- For SMTP:
  - SMTP is used in conjunction with the FAILEDJOB statement. If the MAIL keyword is used on the FAILEDJOB statement, then the SMTP *server_name* should be set to the name of the SMTP server and an optional CLASS and Destination NJE node.
  - When an attempted print job fails and the MAIL keyword used on the FAILEDJOB statement, then the LPD server sends a notice of the failure to the SMTP server. This notice is then forwarded to the user ID that submitted the print request.

- For FAILEDJOB:

  If the MAIL parameter is specified for any service, then the SMTP statement must be included in the LPD configuration data set.

- For EXIT:
  - If the jobname is not specified on the corresponding LPR operation, JOB is the data set name that was printed by LPD.
  - If CLASS is omitted on the LPR operation, it contains the sending system's host name.
  - The following parameters are passed to the *program* but not defined in the EXIT statement.

    param1
    A pointer to a full word return code.

param2
A Pascal string containing the DD name of the spool file, the data set name of the control file, the printer name, and the total number of bytes in the print job. The first two bytes of the Pascal string are the number of bytes of character data starting at byte 3.

param3
A pointer to an open DCB for the JES spool file. The DCB is (DSORG=PS, MACRF=PL, RECFM=UM) for SERVICE printer or SERVICE PUNCH devices. THE DCB is (DSORG=PS, MACRF=PL, RECFM=U) for SERVICE RECFMU devices.

STEPLIMIT Statement

Use the STEPLIMIT statement to limit the number of data files and configuration files allowed in a job received by LPD. Jobs that are too complex will be rejected with a NACK and will not be printed.

Syntax

Parameters

limit

An integer specifying the maximum number of data files and configuration files allowed in a single job received by LPD. When a wildcard is used in the 'filename' with LPR in some systems, the files are combined into one complex job with many data files.

Usage Notes

- Each data file and control file requires a temporary data set on MVS. Each requires memory for control blocks and I/O buffers.
- STEPLIMIT defaults to the recommended value of 80 when the keyword is not specified. Increasing this value may cause memory allocation problems with certain system configurations.
- If LPD runs out of memory, reduce the value of either JOBPACING or STEPLIMIT.

UNIT Statement

Use the UNIT statement to specify the specific type of DASD where LPD will write its temporary data sets while the transfer of data from an LPR client occurs.

Syntax

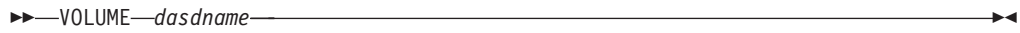
Parameters

dasdname
The generic name of a group of DASD.

VOLUME Statement

Use the VOLUME statement to specify the specific DASD volume where LPD writes its temporary data sets while the transfer of data from an LPR client occurs.

Syntax



Parameters

dasdname
The volume serial number. The value specified for *name* is case-sensitive.

Examples

Set the volume name for new data set to WRKLB4:

```
VOLUME WRKLB4
```

Operating the LPD Server Using the SMSG Interface

The LPD server supports an SMSG interface that allows authorized users to direct commands to the server while it is running.

To use the SMSG interface, you must update the LPD server configuration data set using the OBEY statement to indicate which user IDs are authorized to use the SMSG interface.

Once you have defined the user IDs with the OBEY statement, you can use the following command:

procname
The member name of the cataloged procedure used to start the LPD server.

TRACE ON
Activates full tracing of the processing within the LPD server. Trace output is written to the //SYSDEBUG DD data set.

TRACE OFF
Deactivates tracing. Issuing this command ends all tracing, regardless of the means used to establish the tracing. This command also turns off the minimal tracing activated by the TYPE option.

PRINT WORK
Dumps the current list of jobs waiting to be printed into the //SYSOUT DD data set. The list is prefaced by the string: `WORK QUEUE START` and ended by the string: `WORK QUEUE END`.

Chapter 25. Configuring the OROUTED Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the OROUTED server. It explains OROUTED’s use of the Routing Information Protocol to help you decide if this server is suitable for your network. It also explains Virtual IP Addressing (VIPA) of OROUTED which gives you the ability to include VIPA routes in routing information for propagation. With VIPA routes, routers in the network can route around interface, device and network failures to reach the VIPA addresses at the destination host. Examples for various configurations are given.

Notes:

1. OMPROUTE, which was introduced in eNetwork Communications Server for OS/390 V2R6 and enhanced in V2R7, is the recommended routing daemon. OROUTED will eventually be removed; you will receive ample formal notice of this change.
2. If running Enterprise Extender with ORouteD, see “Configuring ORouteD with Enterprise Extender” on page 940.
3. OROUTED does not support zero subnets.
4. OROUTED does not support equal-cost multipath routes to a destination network or host. However, OROUTED can be used in conjunction with statically defined equal-cost multipath routes in the GATEWAY statement of the TCP/IP profile.

Understanding OROUTED

The route daemon is a server that implements the Routing Information Protocols (RIP) described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). It provides an alternative to the static TCP/IP gateway definitions. When configured properly the MVS host running with OROUTED becomes an active RIP router in a TCP/IP network. The OROUTED server dynamically creates and maintains the network routing tables using RIP. RIP allows gateways and routers to periodically broadcast their routing tables to adjacent nodes. This enables the OROUTED server to update the host routing table. For example, the OROUTED server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists. OROUTED has the following characteristics:-

- Deletion of all RIP routes at startup. The -del start parameter might be used to delete all dynamic routes from the routing table upon initialization of OROUTED.
- OROUTED is an OS/390 UNIX application. It requires the Hierarchical File System (HFS) to run.
- OROUTED can be started from an MVS procedure or from the OS/390 shell command line.
- OROUTED uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.
- All messages and trace information is sent to the syslogd, except for output from the *-d* and *-dp* parameters, which is sent to STDOUT.

- A default mode of operation is for the program to close STDIN, STDOUT and STDERR. This allows for userids to cleanly exit the OS/390 shell after starting the program in the background. As a consequence, the program printf statements are also disabled. The parameter ″-ep″ enables printf's. If this parameter is specified, the program should not be run in the background, because the userid will not be able to exit the shell until the background job has been killed.
- OROUTED supports ATM network interfaces as gateways if the interface to the IP network is operating in ATM LAN Emulation mode. However, OROUTED cannot be used to manage routes on an ATM interface operating in native mode due to limitations. See "Routing Information Protocol (RIP)" for more information.
- OROUTED and OMPROUTE cannot run on the same stack concurrently.
- OROUTED needs to be started by a RACF-authorized user ID.

Routing Information Protocol (RIP)

The Routing Information Protocol (RIP) is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. IGPs are used to manage the routing information of a single autonomous system, or a single piece of the TCP/IP network. RIP has many limitations and is not suited for every TCP/IP environment. Before installing the OROUTED server, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. See "Appendix D. Related Protocol Specifications (RFCs)" on page 1177 for more information about RFC 1058 and RFC 1723.

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*. A gateway is defined as zero hops from directly connected networks, one hop from networks that can be reached through one gateway, and so on. 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.

The OROUTED server propagates routing information to the neighboring gateway's on gateway's directly connected networks every 30 seconds. The server receives updates from neighboring gateways periodically and uses this information to update the routing tables. If an update has not been received from a gateway in 180 seconds (3 minutes), OROUTED assumes the gateway is down and sets all the routes through that gateway to a metric of 16 (infinity). If an update has not been received from a gateway in another 120 seconds (2 minutes), OROUTED deletes all of the routes through that gateway.

During the intervals specified by the *interface.scan.interval* and *interface.poll.interval* values on the OPTIONS statement, OROUTED checks to determine if a local interface is up or down by scanning the TCP/IP interface tables. It also checks to see if an interface has been added or reactivated.

For networks that are not point-to-point, such as Token-Ring and Ethernet, OROUTED receives its own copy of broadcasted or multicasted RIP packets over the interfaces, provided that the interfaces are active. Other networks, such as point-to-point, can be managed by OROUTED as long as there are RIP services managing the other end of point-to-point link(s). If the destination addresses are defined for these point-to-point networks, the RIP packets are unicasted. Otherwise, the RIP packets are broadcasted or multicasted.In networks where there are adjacent routers or hosts not running RIP services, OROUTED will not be receiving updates over the links and eventually will delete all of the routes to these networks.

To prevent the routes to these networks not running RIP services from being deleted, passive routes may be coded in a OROUTED gateways data set. For more information, see “OROUTED Gateways” on page 921 and “OROUTED Parameters” on page 934.

Because OROUTED requires link-level broadcasting or multicasting to send routing updates, it requires routers that do not support link-level broadcasting or multicasting (for example, HYPERchannel) to be active gateways. For more information on how to manage networks without link-level broadcasting or multicasting support, see “Active Gateways” on page 922. By configuring active gateways, the RIP packets are unicasted to the neighboring gateways.

OROUTED supports ATM network interfaces as gateways if the interface to the IP network is operating in ATM LAN Emulation mode. In this case, the ATM network interface is treated as a LAN network interface by OROUTED. However, due to limitations, OROUTED cannot be used to manage ATM network interfaces operating in native mode. In this case, static routes must be configured to the ATM network via the GATEWAY statement in the TCPIP profile. See “GATEWAY Statement” on page 193 for more information on how to add static routes to ATM networks. Also, to prevent OROUTED from adding or deleting routes to the ATM network through other interfaces, it is necessary to add blocking statements in the OROUTED gateways file for all other interfaces on the MVS system. All IP subnet addresses in the ATM network should be blocked for all gateways on the MVS system. For more information on blocking routes, see “RIP Input/Output Filters” on page 920.

RIP Version 2

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags to provide EGP-RIP and BGP-RIP interactions
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 or another IGP. OROUTED will not generate route tags, but will preserve them in received routes and readvertise them when necessary.

Variable subnetting support
Variable-length subnet masks are being included in routing information so that dynamically-added routes to destinations outside subnetworks or networks can be reachable.

Immediate Next Hop for shorter paths
Next hop IP addresses, whenever applicable, are being included in the routing information. Its purpose is to eliminate packets being routed through extra hops in the network. OROUTED will not generate immediate next hops, but will preserve them if they are included in the RIP packets.

Multicasting for RIPv2 packets to reduce load on hosts
An IP multicast address 224.0.0.9, reserved for RIPv2 packets, is used to reduce unnecessary load on hosts which are not listening to RIPv2 messages. RIPv2 multicasting is dependent on interfaces that are multicast-capable. By default, RIPv1 packets will be broadcast for interfaces that are not multicast-capable.

Authentication of RIPv2 packets for routing update security
Authentication keys, consisting of passwords, can be configured to be included in the outgoing RIPv2 packets for authentication by adjacent routers as a routing update security protection. Likewise, incoming RIPv2

packets are checked against local authentication keys. Any outgoing or incoming RIPv1 packets are not authenticated. For maximum security, configure OROUTED such that it will supply and receive RIPv2 packets only in addition to specification of authentication keys. The authentication keys are configurable on a router-wide or per-interface basis.

Configuration switches for RIPv1 and RIPv2 packets
Configuration switches are provided to selectively control which versions of RIP packets are to be sent or received over network interfaces. The switches should be configured based upon the routing capabilities of the network and are configurable on a router-wide or per-interface basis.

Supernetting support
The supernetting feature is part of Classless InterDomain Routing (CIDR) function. Supernetting provides a way to combine multiple network routes into fewer 'supernet' routes. This means that the number of network routes in the routing tables becomes smaller for advertisements. Supernet routes are received and sent in RIPv2 messages. If local supernet routes are defined for OROUTED, they will be advertised to adjacent routers. Local supernet routes are generated by OROUTED for interfaces with subnet masks that are less than the network class mask in value.

OROUTED Miscellaneous Features

OROUTED supports the following miscellaneous features:

- Multiple Network Attachments

  Multiple attachments and IP addresses on the same network are supported, providing redundant paths to other hosts or routers on directly-attached networks. For more information, see "Multiple Network Attachments Support" on page 104.
- Virtual IP Addressing (VIPA)

  If virtual IP addresses are configured on the OS/390 server and there are multiple network attachments, OROUTED can be used to advertise the VIPA routes to the network, providing the necessary routing information to the adjacent routers or hosts running RIP services. Using the VIPA routes, the adjacent routers or hosts will be able to reach the VIPA addresses as the destinations and to route around failures for fault tolerance support. For more information, see "Chapter 3. Virtual IP Addressing" on page 71.

RIP Input/Output Filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by OROUTED and consist of:

1. Route Blocking (or NoReceiving)
2. Route Forwarding (Unconditional and Conditional)
3. Route Receiving (Unconditional and Conditional)
4. Route NoForwarding
5. Interface Supply Switch
6. Interface RIP On/Off Switch
7. Default Route Only Supply Switch
8. Virtual Route Only Supply Switch
9. Default and Virtual Routes Only Supply Switch

10. Local (directly-connected) Routes Only Supply Switch
11. Triggered Updates Only Supply Switch
12. Gateway NoReceiving

For more information on these RIP input/output filters, see the OROUTED procedure parameters in “OROUTED Parameters” on page 934 and the options statement in “Step 6: Configure the Gateways File or Data Set (Optional)” on page 928.

RIP Routes

Dynamic routes are any routes created by the following dynamic routing applications:

- OMPROUTE (OSPF/RIP)
- OROUTED (RIP)
- NCPROUTE (RIP)
- ICMP Redirects
- Other (created by customer application)

However, RIP routes are dynamic routes created only by RIP applications, like OROUTED and OMPROUTE.

To help maintain the integrity of the routing table, at initialization, OROUTED attempts to delete all RIP routes from the stack routing table and then adds RIP routes based on BSDROUTINGPARMS and the optional gateways file. The -del start parameter can be used to delete all dynamic routes from the routing table upon initialization of OROUTED.

When OROUTED terminates, RIP routes are not deleted. If you want to remove all RIP routes upon OROUTED termination, used the -kdr MODIFY option.

OROUTED Gateways

Passive RIP Routes

Passive RIP routes are known by both TCP/IP and OROUTED. Information about passive routes is put in TCP/IP’s and OROUTED’s routing tables. A passive entry in OROUTED’s routing table is used as a placeholder to prevent a route from being propagated and from being overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router. Using passive routes can create routing loops, so they need to be created carefully.

Defining passive routes such as these should be avoided:

A to C is via B.
B to C is via A.

Passive routes should be used when adding routes where the host/net is not running RIP. Passive routes should also be used when adding a default route, since this is the only way to prevent a route from timing out.

External RIP Routes

External RIP routes are known by OROUTED, but not by TCP/IP. External routes, such as the External Gateway Protocol (EGP), are managed by other protocols. The OROUTED server needs to know not to interfere with these and not to delete them.

An external entry exists in OROUTED's routing tables as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. OROUTED does not manage external routes. Therefore, OROUTED only knows that there is an existing route to host/net and one that is known to TCP/IP.

External routes should be used when the local host is running with some type of non-RIP routing protocol which dynamically changes the TCP/IP routing tables. The foreign host does not need to run any routing protocol, since the only concern is how to route traffic from the local host to the foreign host, and how to prevent multiple routing protocols from interfering with each other.

Active Gateways

Active gateways are treated as remote network interfaces. Active gateways are routers which are running RIP, but are reached by a medium which does not allow broadcasting or multicasting and is not point-to-point. OROUTED normally requires that routers be reachable via broadcast for non-point-to-point links or via unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to OROUTED's interface list. OROUTED will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non-broadcast and non-point-to-point network, and is directly connected to the local host.

OROUTED will communicate with active routers by unicast transmissions to the gateway address. Routes are not added to either OROUTED or the TCP/IP routing table immediately. They are added and propagated normally when route advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast or multicast communication on non-point-to-point links. Interfaces which are not broadcast-capable or multicast-capable, not point-to-point, and are not active gateways are assumed to be loopback interfaces to the local host. Also, while a route to an active gateway might time out, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

OROUTED Gateways Summary

Table 42 provides a summary of the OROUTED gateways and their characteristics.

Table 42. OROUTED Gateways Summary

	Propagated?	Known by TCP/IP?	Known by OROUTED?	Timeout?
Dynamic (1)	Yes	Yes	Yes	Yes
Passive	No (2)	Yes	Yes	No

Table 42. OROUTED Gateways Summary (continued)

	Propagated?	Known by TCP/IP?	Known by OROUTED?	Timeout?
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes
Dynamic routing provided by OROUTED.				

Notes:

1. Except directly-connected passive routes.
2. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in the HOME list.

Configuration Process

The steps to configure OROUTED are as follows:

1. Configure the OROUTED profile.
2. Update PORT, BSDROUTINGPARMS, GATEWAY, and IPCONFIG statements in the TCPIP profile.
3. Update the resolver configuration file.
4. Update the OROUTED cataloged procedure (optional).
5. Specify the OROUTED port number in the SERVICES file or data set.
6. Configure the gateways file or data set (optional).

   Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways file or data set.
7. If not already started, configure and start syslogd.
8. RACF-authorize user IDs for starting OROUTED.

Step 1: Configure the OROUTED Profile

OROUTED supports sending and receiving both RIP version 1 and RIP version 2 packets. A configuration file, the OROUTED profile, determines the mode of operation. The following is the search order used to locate the OROUTED configuration data set or file:

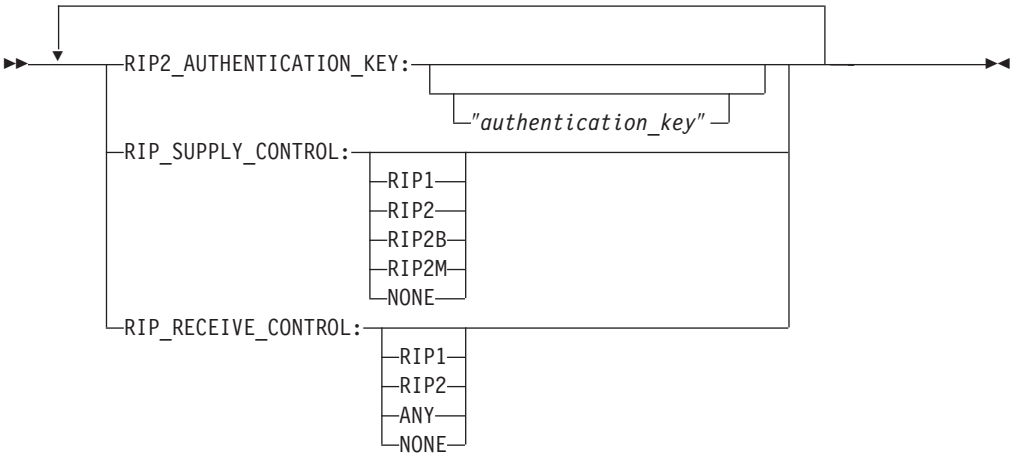
1. If the environment variable ROUTED_PROFILE has been defined, OROUTED uses this value as the name of an MVS data set *(//'mvs.dataset.name')* or HFS file *(/dir/subdir/file.name)* to access the profile.
2. /etc/routed.profile
3. *hlq*.ROUTED.PROFILE

The following are the syntax rules for the OROUTED profile:

- Keywords can be specified in mixed case.
- Blanks and comments are supported in the OROUTED profile. Comments are identified by a semicolon in any column.
- Profile statements may start in any column; however, wrapping to the next record for continuation is not allowed.
- There should be no sequence numbers in the data set or file.

A sample profile is provided in *hlq*.SEZAINST(EZARTPRF).

The following are the options that can be included in the OROUTED profile:



RIP2_AUTHENTICATION_KEY:
A constant. The value that follows is the authentication key.

authentication key
Specifies a plain text password containing up to 16 characters. The authentication key must be enclosed in double quotes. The key is used on a server-wide basis and can contain mixed case and blank characters. The key will be used to authenticate RIP Version 2 packets and be included in the RIP responses for authentication by adjacent routers running RIP Version 2. A null key (either no key is specified, or "" is specified) indicates that authentication is disabled. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets.

RIP_SUPPLY_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP supply control for all interfaces. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2 Unicast/Multicast RIP Version 2 packets

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

NONE Disable sending RIP packets

RIP_RECEIVE_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP receive control for all interfaces. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

Figure 41 on page 925 is a sample OROUTED configuration file:

```
; EZARTPRF
;
; COPYRIGHT = NONE.
;
; Sample OROUTED profile.
; See the "IP Configuration Guide for more information"
;
; ---------------------------------------------------------------------
; RIP_SUPPLY_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1  - Unicast/Broadcast RIP Version 1 packets (Default)
; 2) RIP2B - Unicast/Broadcast RIP Version 2 packets (Not Recommended)
; 3) RIP2M - Unicast/Multicast/Broadcast RIP packets (Migration)
; 4) RIP2  - Unicast/Multicast RIP Version 2 packets
; 5) NONE  - Disables sending RIP packets
;
; Note: If RIP2 is specified, the RIP Version 2 packets are multicast
;       over multicast-capable interfaces only.  No RIP packets are
;       sent over multicast-incapable interfaces.  For RIP2M, the RIP
;       Version 2 packets are multicast over multicast-capable
;       interfaces and RIP Version 1 packets are broadcast or unicast over
;       multicast-incapable interfaces.  For RIP2B, the RIP Version
;       2 packets are broadcast or unicast; this option is not recommended since
;       host route misinterpretations by adjacent routers running RIP
;       Version 1 can occur.  For this reason, RIP2B may become
;       obsolete in a future release.  For point-to-point interfaces
;       that are non-broadcast and multicast-incapable, the RIP
;       Version 2 packets are unicast.

RIP_SUPPLY_CONTROL: RIP1

; RIP_RECEIVE_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1   - Receive RIP Version 1 packets only
; 2) RIP2   - Receive RIP Version 2 packets only
; 3) ANY    - Receive any RIP Version 1 and 2 packets (Default)
; 4) NONE   - Disables receiving RIP packets

RIP_RECEIVE_CONTROL: ANY

; RIP2_AUTHENTICATION_KEY specifies a plain text password containing
; up to 16 characters.  The authentication key must be enclosed in
; double quotes.  The key is used on a server-wide basis and can
; contain mixed case and blank characters.  The key will be used to
; authenticate RIP Version 2 packets and be included in the
; RIP responses for authentication by adjacent routers running RIP
; Version 2.  A null key (either no key is specified or "" is
; specified) indicates that authentication is disabled.  For
; maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL
; to RIP2.  This will discard RIP1 and unauthenticated RIP2 packets.

RIP2_AUTHENTICATION_KEY:
```

Figure 41. Sample OROUTED Configuration File

Step 2: Update Configuration Statements in PROFILE.TCPIP

To ensure that UDP port 520 is reserved for OROUTED, also add the name of the member containing the OROUTED cataloged procedure to the PORT statement in PROFILE.TCPIP:

```
PORT
   520 UDP OROUTED
```

In addition, configure the BSDROUTINGPARMS statements with your routing information. The GATEWAY statement is not used and should be removed or commented from PROFILE.TCPIP.

Code the following on the IPCONFIG statement in PROFILE.TCPIP:

```
IPCONFIG IGNOREREDIRECT DATAGRAMFWD
```

Do not specify the no forwarding (NOFWD) option. To enable variable subnetting, add the VARSUBNETTING option. If OROUTED is to be used to either send or receive RIP version 2 packets, the VARSUBNETTING option must be specified in the TCPIP profile. To enable outbound source VIPA, add the SOURCEVIPA option.

See “Chapter 4. Configuring the TCP/IP Address Space” on page 97 for descriptions and examples of these statements.

Note: If you want to be able to start OROUTED from the OS/390 shell, use the special name OMVS as follows:

```
PORT 520 UDP OMVS
```

This enables the entire “OMVS job group” (that is, all OS/390 shell users). Only RACF-authorized users can start OROUTED.

Step 3: Update the Resolver Configuration File

The resolver configuration file or data set contains keywords that are used by OROUTED. Two important keywords in the resolver file are DATASETPREFIX and TCPIPjobname. The value assigned to DATASETPREFIX will determine the high-level qualifier (*hlq*). The *hlq* is then used in the search order for other configuration files. If no DATASETPREFIX keyword is found in the resolver configuration data set or file, a default of TCPIP is used. In a CINET environment, the value assigned to TCPIPjobname will be used as the name of the stack with which OROUTED attempts to establish a connection. In an INET environment, it is not necessary to set TCPIPjobname, but if it is set, it must be set to ″INET″.

For a description of the search order used by the resolver to locate the resolver configuration data set or file, see “Search Order and Configuration Files for TCP/IP Applications” on page 19.

Step 4: Update the OROUTED Cataloged Procedure (Optional)

If OROUTED is to be started by a procedure, update the cataloged procedure OROUTED by copying the sample in *hlq*.SEZAINST(OROUTED) to your system or recognized PROCLIB. Specify OROUTED parameters and change the data set names as required to suit your local configuration.

Note: When using PGM=BPXBATCH to start OROUTED, STDOUT and STDERR cannot be directed to any SYSOUT class; they must be directed to an HFS file.

OROUTED Cataloged Procedure

A data set or file may be used to set the environment variables for an invocation of OROUTED. This file is specified on the STDENV statement. An example of its contents is included in the OROUTED cataloged procedure sample. For more complete information about STDENV, refer to *OS/390 UNIX System Services User's Guide*.

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZBRDORP
//*
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        5645-001 5655-HAL (C) Copyright IBM Corp. 1996.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//*        OROUTED can be started with a variety of parameters.
//*        In this example, the "-ep" parameter enables displaying
//*        program output to standard out, and the "-t -t" parameter
//*        enables 2 levels of tracing.
//*
//OROUTED  PROC
//OROUTED  EXEC PGM=OROUTED,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/-ep -t -t')
//*
//*        The stdout stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stdout file
//*        every time OROUTED is started.  If you want to retain
//*        previous stdout information, change it to OAPPEND.
//*
//SYSPRINT DD SYSOUT=*
//*SYSPRINT DD PATH='/tmp/orouted.stdout',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        The stderr stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stderr file
//*        every time OROUTED is started.  If you want to retain
//*        previous stderr information, change it to OAPPEND.
//*
//SYSOUT   DD SYSOUT=*
//*SYSOUT   DD PATH='/tmp/orouted.stderr',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        OROUTED can use certain environmental variables, such
//*        as NLSPATH (to determine the location of the message
//*        catalog), RESOLVER_CONFIG (to determine the location
//*        of the file that contains parameters TCPIPjobname and
//*        DATASETPrefix), ROUTED_PROFILE (to determine the
//*        location of the optional OROUTED profile), and
//*        GATEWAYS_FILE (to determine the location of the
//*        optional gateways file).  Examples of the contents of
//*        this file are as follows:
//*
//*             RESOLVER_CONFIG=/etc/resolv.conf
//*             ROUTED_PROFILE=//'SAMPLE.ROUTED.PROFILE'
//*
```

```
//*        Define STDENV with the name of the file that contains
//*        the environmental variables to be used for this
//*        invocation of OROUTED.
//*
//STDENV   DD PATH='/etc/orouted.env'
//*STDENV   DD DSN=SAMPLE.OROUTED.ENV,DISP=SHR
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Step 5: Specify the OROUTED Port Number in the SERVICES File

The services data set or file contains the relationship between service names (servers) and port numbers in the OS/390 UNIX environment. The portion of the services file relevant to OROUTED is shown in Figure 42. The data set or file must exist for OROUTED to run. The following search order is used to find the services data set or file:

1. /etc/services
2. *userid*.ETC.SERVICES, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq*.ETC.SERVICES

```
# Start of IBM added services ...
route        520/udp       router routed
```

Figure 42. Sample Portion of Services File

Step 6: Configure the Gateways File or Data Set (Optional)

The OROUTED server queries the network and dynamically builds routing tables from routing information transmitted by other routers that are directly connected to the network. The gateways file or data set is used to further configure the routing tables.

Note: The gateways file or data set is not related to the GATEWAY statement used in the PROFILE.TCPIP data set.

The OROUTED server uses the following search order to locate the GATEWAYS configuration data set or file:

1. If the environment variable GATEWAYS_FILE has been defined, OROUTED uses this value as the name of an MVS data set (//'mvs.dataset.name') or HFS file (/dir/subdir/file.name) to access the gateways file
2. /etc/gateways
3. *hlq*.ETC.GATEWAYS

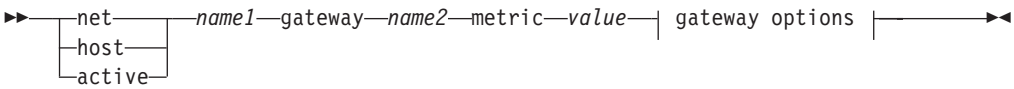
A sample gateways file is provided in *hlq*.SEZAINST(EZARTGW).

A `passive` entry in the gateways file or data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways file or data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

Syntax Rules

- Keywords can be specified in mixed case.
- Blanks and comments are supported in the gateways file or data set. Comments are identified by a semicolon in column 1.
- There should be no sequence numbers in the data set.

The syntax for the gateways file or data set is:



gateway options:

net
Indicates the route goes to a network.

host
Indicates the route goes to a specific host.

active
Indicates that the route to the gateway will be treated as a network interface. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

name1
Can be either a symbolic name or the IP address of the destination network or host. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal nor octal notation will be accepted.

name1 must be specified as “active” if this is for an active gateway. The last entry in the data set must specify an active gateway.

gateway
A constant. The parameters that follow this keyword identify the gateway or router for this destination.

name2
Can be either a symbolic name or the IP address of the gateway or router for this destination. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal nor octal notation will be accepted.

metric
A constant. The value that follows this keyword is the hop count to the destination host or network.

value
The hop count to this destination. This number is an integer in the range of 0 through 16, where 16 (infinity) indicates the network cannot be reached.

passive
A passive gateway does not exchange routing information. Information about the passive gateway is maintained in the local routing tables indefinitely and is only local to this OROUTED server. Passive gateway entries for indirect routes are not included in any routing information that is transmitted. Directly connected passive routes are included.

external
An external gateway parameter indicates that entries for this destination should never be added to the routing table. The OROUTED server discards any routes for this destination that it receives from other routers. Only the destination field is significant. The gateway and metric fields are ignored.

active
Active gateways are treated as network interfaces. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

mask
A constant. The value that follows this keyword is the subnet mask for the route.

subnetmask
A bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous in the network portion of the *subnetmask*. If the *subnetmask* is not specified, OROUTED will default the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, then the network class mask for the route is used.

Note: For more information on passive, external, and active gateways, see "OROUTED Gateways" on page 921.

The following example shows the contents of a gateways file or data set containing multiple entries:

```
net    acmenet       gateway  gateway.acme.com  metric 5  passive
host   vm3.ibm.com   gateway  9.67.43.126       metric 6  passive
host   bad.host      gateway  xxx               metric 1  external
active active        gateway  9.3.1.110         metric 3  active
net    0.0.0.0       gateway  9.67.112.1        metric 1  passive
```

In the first entry, the route indicates that `acmenet` can be reached through the gateway `gateway.acme.com`, and that it is 5 hops away.

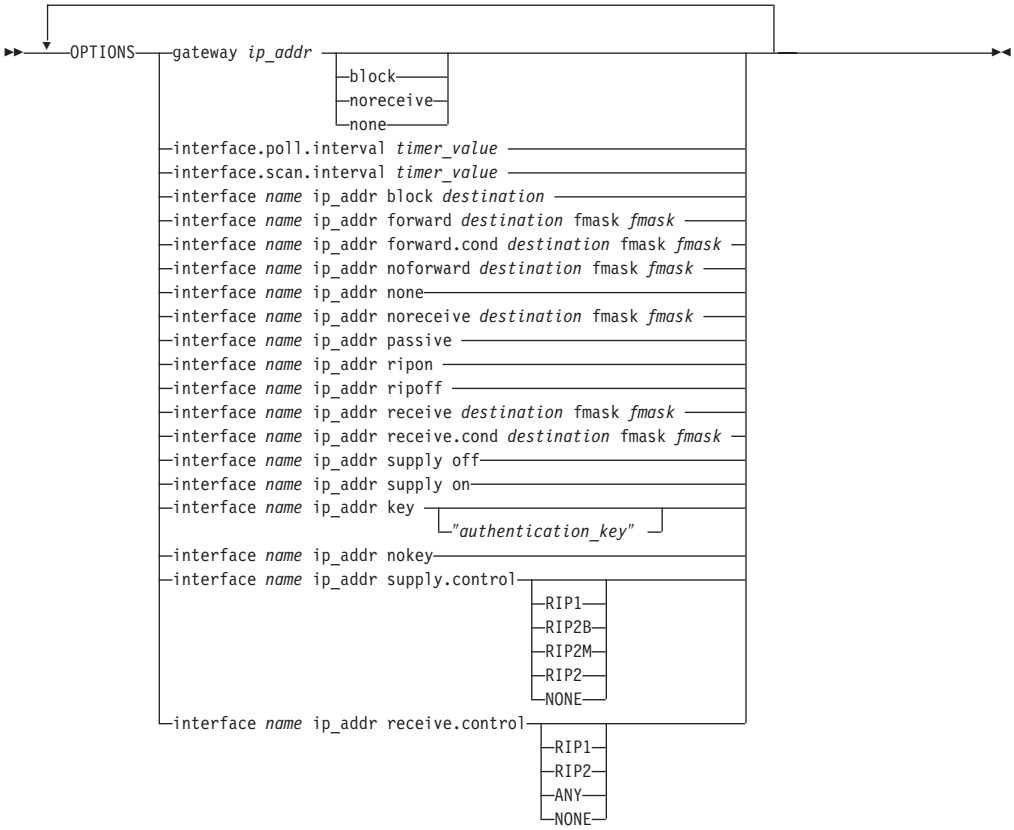
In the second entry, the route indicates that `vm3.ibm.com` can be reached through the gateway `9.67.43.126`, and that it is 6 hops away.

In the third entry, the external gateway parameter indicates that routes for the host `bad.host` should not be added to the routing tables, and that routes received from other OROUTED servers for `bad.host` should not be accepted.

The fourth entry shows an active gateway.

The fifth entry shows a default route to the destination gateway 9.67.112.1.

The syntax for the OPTIONS statement for the gateways file or data set is:



gateway
A constant. The value that follows this keyword identifies the gateway or router.

interface.scan.interval
Specifies the time interval in seconds for the interface scan interval. OROUTED uses this timer value to rescan existing interfaces for up/down status, new interfaces, and new HOME lists. New interfaces and HOME lists are dynamically added using VARY TCPIP,,CMD=OBEYFILE commands.
timer_value
The range is from 30 to 180 seconds in multiples of 30 seconds. The default is 60 seconds.

interface.poll.interval
Specifies the time interval in seconds for the interface poll interval. OROUTED uses this timer value to check existing interfaces for up/down status only. Triggered updates are issued during interface outages to inform adjacent routers of unreachable routes so that alternative routes can be discovered.

timer_value
The range is from 15 to 180 seconds in multiples of 15 seconds. The default is 30 seconds.

interface
A constant

name
Specifies the name of the interface as used in the HOME list. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all interface names.

ip_addr
Specifies the internet address of the interface associated with the interface name. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all internet addresses of the interfaces.

block
For the interface option, specifies that the *destination* route in the received broadcasts for this interface is to be ignored. For the gateway option, specifies that routing table RIP responses from this gateway are to be ignored. This option is provided as a RIP input filter.

destination
Specifies the destination route in network, subnetwork, or host format. A specification of an asterisk (*) indicates that all destination routes to be used with the noforward and noreceive options. This serves as a ″blackhole″ filter option which can be used to filter out all routes from RIP packets to be sent or received over an interface and allow routes with specified forward and receive filters to be used.

fmask
Specifies the optional route filter mask.

forward
Specifies that the *destination* route in the RIP responses is to be forwarded to this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

forward.cond
Specifies that the *destination* route is to be forwarded to this interface only when the interface is active. In case of an interface outage, OROUTED will include the *destination* route in the RIP responses to other active interfaces. After recovery of an interface outage, ORouteD will resume to sending the destination route over this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

noforward
Specifies that the destination route in the RIP responses is not to be forwarded. This option is provided as a RIP output filter.

noreceive
See description for block.

passive
Same as ripoff.

receive
Specifies that the *destination* route is to be received over this interface only. If it is received over any other interface, the route is discarded. This option is provided as a RIP input filter.

receive.cond
Specifies that the *destination* route is to be received over this interface only when the interface is active. In case of an interface outage, OROUTED will allow the *destination* route in the RIP responses to be received over other active interfaces. This option is provided as a RIP input filter and can be used for inbound and outbound traffic splitting..

ripoff
Specifies that RIP is disabled for this interface. OROUTED will not send or receive RIP updates. This option is provided as a RIP input and output filter.

ripon
Specifies that RIP is enabled for the interface. This is the default for all interfaces. This option should be used when RIP has been previously disabled for an interface with the ripoff option, but is now required to be enabled for that interface.

supply off
Specifies that supplying RIP responses is disabled for this interface. OROUTED will not send, but continues to receive RIP responses. This option is provided as a RIP output filter.

supply on
Specifies that supplying RIP responses is enabled for this interface. This option is provided as a RIP output filter.

none
For the interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If an asterisk (*) is specified for the interface *name* and *ip_addr*, all options will be cleared from all interfaces. For the gateway option, specifies that any RIP filter options for this gateway are to be turned off or reset. If an asterisk (*) is specified for the internet address, all gateway entries with gateway options will be cleared.

key
Specifies a plain text password containing up to 16 characters for the authentication key to be used for this interface and is used to override the router-wide setting defined in the OROUTED profile data set. The key must be enclosed in double quotes for the delimiters and can contain mixed case and blank characters. A no key or null key (″″) specification indicates that the router-wide key will be used as the default.

authentication_key
An authentication key containing up to 16 characters to be used for this interface and is used to override the OROUTED profile setting. The key must be enclosed in double qotes. The key will start with the first character past the first quotation mark and end at the last character before the last quotation mark on the line.

nokey
Specifies that authentication is disabled for this interface even though the router-wide specification from the OROUTED profile is defined.

supply.control
A constant. Specifies that the keyword follwing is to be used as the RIP supply control for this interface and is used to override the OROUTED profile setting. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

RIP2 Unicast/Multicast RIP Version 2 packets

NONE Disable sending RIP packets

receive.control
A constant. Specifies that the variable follwing is to be used as the RIP receive control for this interface and is used to override the OROUTED profile setting. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

The following example shows the options entries of a gateways file or data set:

```
options interface.scan.interval 90
options interface.poll.interval 15
options interface ETH1 10.1.1.1 passive
options interface ETH1 10.1.1.1 supply off
options interface TR1 9.67.112.25 forward 11.0.0.0
options interface TR1 9.67.112.25 forward.cond 12.0.0.0
options interface TR1 9.67.112.25 block 9.1.0.0
options interface TR1 9.67.112.25 supply.control rip1
options interface ETH1 10.1.1.1 receive.control rip2
options interface ETH1 10.1.1.1 key
options interface CTC0  9.67.114.22 key "shredder"
options interface ETH1 10.1.1.1 none
options interface * * none
options gateway 9.2.1.4 noreceive
options gateway 9.2.1.4 none
options gateway * none
```

Step 7: Configure and Start syslogd

If not already started, configure and start syslogd. For more information on syslogd, see “Appendix E. Syslog Daemon” on page 1183. Otherwise, all output messages will go to the operator system console.

Step 8: RACF-Authorize User IDs

To control which users can start OROUTED (and thus reduce risk of an unauthorized user starting it and affecting the contents of the routing table), the appropriate users must be RACF-authorized to the entity MVS.ROUTEMGR.OROUTED. To do this, the following commands must be entered from a RACF user ID, substituting the authorized user ID on the ID (userid) parameter:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OROUTED) UACC(NONE)
PERMIT MVS.ROUTEMGR.OROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

OROUTED Parameters

OROUTED accepts the command line parameters listed below. These parameters are valid when starting the program from either an MVS procedure or from the OS/390 shell. They are also valid when modifying OROUTED with the MODIFY command. For information on using the MODIFY command, see“Controlling OROUTED with the MODIFY Command” on page 945.

-del
All dynamic routes are deleted from the routing table upon initialization of OROUTED. By default, OROUTED deletes only RIP routes at startup.

-d
Enables printing internal debug information to STDOUT. This option should only be used to debug problems. When this option is specified, the -ep parameter is set internally.

-dp
Traces packets to and from adjacent routers and received and sent RIP packets. Packets are displayed in data format. Output is written to STDOUT.

-ep
Enable display of program print statements to STDOUT and STDERR. Information can be saved to a file by redirecting STDOUT to a file using the '>' operator. If this option is specified, and the program is started in the background from an OS/390 shell, the userid will not be able to exit the shell until the program has ended.

-g
Enables the default router. When this option is specified, OROUTED will add a default route to its routing information and propagate it over all local interfaces. If the adjacent routers add the default route to their routing tables, OROUTED will receive all unknown packets from them and funnel them to a destination router, provided that a default route is defined. If you use this option, we recommend that you define a default route to a destination router in the gateways file or data set. See "Configuring a Default Route" on page 940.

Note: Do not use this option if default routes are to be learned dynamically from adjacent routers.

-h Include host routes in addition to network routes for the RIP responses. Adjacent routers must be able to receive host routes to prevent NETWORK UNREACHABLE problems from occurring.

-hv
Include only VIPA host routes in addition to network routes for the RIP responses. Adjacent routers must be able to receive host routes' otherwise, network or subnetwork portions of VIPA addresses must be unique for each OS/390 TCP/IP stack.

-q
Suppresses supplying routing information.

-sd
Supply default route only. When this option is specified, the -g parameter is set internally. This option is provided as a RIP output filter.

-sdv (or -svd)
Supply network-specific VIPA routes and default routes only. See parameter descriptions for -sv and -sd. This option is provided as a RIP output filter.

-sl Supply local (directly-connected) routes only. This option is provided as a RIP output filter.

-st
Supply triggered updates only. Similar to the -q parameter except that OROUTED will supply network unreachable routing information during interface outages so that adjacent routers can recover by switching to different routes rather than relying on three-minute timeouts. This option is provided as a RIP output filter.

-sv
Supply network-specific VIPA routes only. Recommended usage is when

multiple network adapters in a OS/390 TCP/IP stack are in the same network; otherwise, network connectivity problems will occur. This option is provided as a RIP output filter.

-svd
Similar to -sdv parameter.

-svh (or -shv)
Supply VIPA (network-specific and host) routes only. This option is provided as a RIP output filter.

-t Activates tracing of actions by the OROUTED server.

-t -t
Activates tracing of actions and packets sent or received.

-t -t -t
Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced and are displayed whenever an interface becomes inactive.

-t -t -t -t
Activates tracing of actions, packets sent or received, packet history, and packet contents. The packet displays the RIP network routing information.

Table 43 shows how the above parameters affect the advertising algorithm for routes in RIP responses to adjacent routers. The parameters can be used as router-wide RIP output filters. To configure interface-wide RIP input and output filters, see the OPTIONS statement in the GATEWAYS configuration data set or file.

Table 43. ORouteD Parameters

Parameter	Host Routes	VIPA Host Routes	Network Routes	VIPA Network Routes	Advertise as Default Router	Local Routes	Unreachable Routes
-g			Yes	Yes	Yes	Yes	Yes
-h	Yes	Yes	Yes	Yes		Yes	Yes
-hv		Yes		Yes			Yes
-s			Yes	Yes		Yes	Yes
-sd					Yes		Yes
-sl						Yes	Yes
-sq or -q							
-st							Yes
-sv				Yes			Yes
-svd				Yes	Yes		Yes
-svh		Yes		Yes			Yes
None			Yes	Yes		Yes	Yes

Specifying Parameters

If OROUTED is to be started from an MVS procedure, add your parameters to `PARM='PGM` in the OROUTED cataloged procedure. For example: `//PARM='PGM /usr/sbin/orouted -g -q -t -t -ep'`

If OROUTED is to be started from an OS/390 shell command line, enter the parameters on the OS/390 shell command line.

For either method of starting OROUTED, the following apply:
- Each parameter is separated by a blank
- Parameters can be specified in mixed case.

Starting OROUTED

In a CINET environment, OROUTED will attempt to connect to a stack name whose name is determined by the *TCPIPJobname* keyword from the resolver configuration data set or file. The *TCPIPJobname* must match the NAME field for ENTRYPOINT(EZBPFINI) in the BPXPRMxx member you used to start OMVS. For information on configuring mulitple TCP/IP instances as OS/390 UNIX CINET physical file systems, see “Considerations for Multiple Instances of TCP/IP” on page 57.

In configurations with multiple stacks, a copy of OROUTED must be started for each stack that requires OROUTED services. To associate OROUTED with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique *TCPIPjobname*. A unique gateways file can be associated with each copy of OROUTED by defining the environment variable GATEWAYS_FILE. In the example in Figure 43 for the OS/390 shell, there are two active stacks with the names TCP_A and TCP_B. First the environment variable RESOLVER_CONFIG is set to point to the file that defines the *TCPIPjobname* TCP_A, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_a, and then OROUTED is started for that stack. Next, RESOLVER_CONFIG is set to point to another configuration file that defines *TCPIPjobname* TCP_B, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_b, and then OROUTED is started for that stack.

```
#  export RESOLVER_CONFIG=/u/user105/tcp_a.conf          !point to TCP_A resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_a              !point to TCP_A gateways file
#  export _BPX_JOBNAME=RDA                               !set procname to RDA
#  export ROUTED_PROFILE=/u/user105/rda.profile          !set routed profile
#  orouted&                                              !start orouted in the background
#  export RESOLVER_CONFIG=/u/user105/tcp_b.conf          !point to TCP_B resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_b              !point to TCP_77 bB gateways file
#  export _BPX_JOBNAME=RDB                               !set procname to RDB
#  export ROUTED_PROFILE=/u/user105/rda.profile          !set routed profile
#  orouted&                                              !start orouted in the background
```

Figure 43. Example Commands to Start Multiple Copies of OROUTED

When running from an MVS procedure, the environment variables can be set by using the STDENV DD statement in the procedure used to start OROUTED. For an example of using the STDENV DD statement, see “OROUTED Cataloged Procedure” on page 927.

Configuration Examples

This section contains examples for configuring the OROUTED server. The following example illustrates a OROUTED configuration.

Configuring a Passive Route

In Figure 44, assume that your OS/390 server is host1 and is running an OROUTED server. The other two hosts, host2 and host3, are not running a RIP server. Your OROUTED server does not learn a route to host3, because host2 is not running a RIP server. Your OROUTED server sends routing updates to host3 over the link to host2 but never receives a routing update from host2. After 180 seconds, your OROUTED server deletes the route to host2. This problem is inherent to the RIP protocol and cannot be prevented.

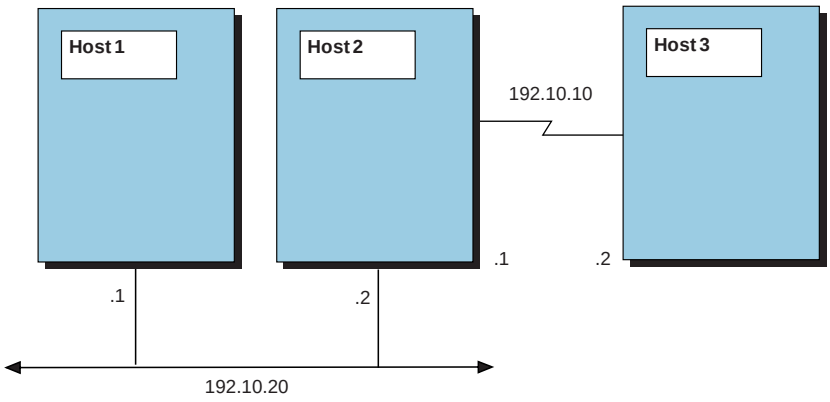


Figure 44. OROUTED Configuration Example

To solve the problem, you should add a passive route to this host in the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, if host2 is not running a RIP server, you can define a directly-connected passive route as follows:

```
host  host2        gateway  host1        metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the HOME list.

Assume that your OS/390 server is now host2 and is running a OROUTED server. host1 is also running a RIP server, but host3 is not. Your OROUTED server sends routing information updates to host3 over the link to host3 but never receives a routing update from host3. After 180 seconds, your OROUTED server deletes the route to host3.

You should add a passive route to this host as follows:

```
host  host3        gateway  host2        metric 1  passive
```

host1 cannot reach host3 unless a passive routing entry is added to host1. For example:

```
host   host3    gateway  host2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an External Route

In Figure 44, assume that your OS/390 server is again host1, which is running an OROUTED server. The other two hosts, host2 and host3, are also running RIP servers. Your OROUTED server normally learns a route to host3 from host2, because host2 is running a RIP server. You might not want host1 to route to host3 for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your OROUTED server from adding a route to host3, add an external route to the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  external

host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an Active Gateway

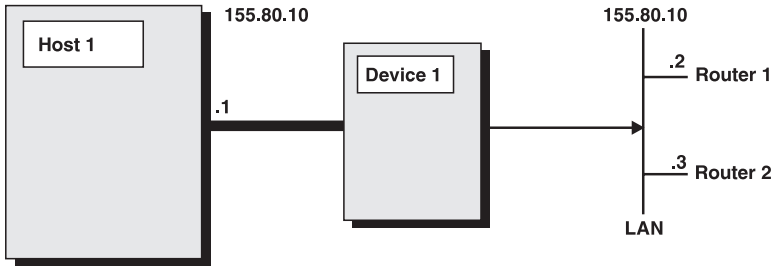


Figure 45. Configuring an Active Gateway

As shown in Figure 45, assume that your MVS host is host1, which is running an OROUTED server and that device1 is a network attachment device that does not support link-level broadcasting or multicasting or one that does not support ARP processing (for example, HYPERchannel). Also, assume that host1 is channel-connected to device1 as a hyperchannel device and that there are routers router1 and router2 on the local area network. Because the IP addresses for router1 and router2 are unknown by host1, they have to be manually configured in host1 for OROUTED to communicate with them. Configuring active gateways for router1 and router2 as remote network interfaces enables OROUTED to send RIP responses to the target addresses.

Include the following definitions in *hlq*.PROFILE.TCPIP for host1, router1, and router2:

1. Specify DEVICE and LINK statements for the hyperchannel. For example:

   ```
   DEVICE HYPER1   HCH    CE2
   LINK   HYPER1A  HCH    2     HYPER1A
   ```

2. Add the TRANSLATE statement for the remote routers on the local area network attached to the hyperchannel device:

   ```
   TRANSLATE 155.80.10.2 HCH    HYPER1A
   TRANSLATE 155.80.10.3 HCH    HYPER1A
   ```

3. Add the hyperchannel link to the HOME statement for the assignment of local IP address:

```
HOME
     155.80.10.1  HYPER1A
```

4. Add the hyperchannel link to the BSDROUTINGPARMS statement:

```
BSDROUTINGPARMS false
      HYPER1A       16384      0      255.255.240.0    0
ENDBSDROUTINGPARMS
```

Define active gateways for the remote routers in the OROUTED gateways file or data set:

```
active active gateway 155.80.10.2 metric 1 active
active active gateway 155.80.10.3 metric 1 active
```

From these active gateway addresses, OROUTED will use them as the destination addresses to send RIP responses to the remote routers. In addition, OROUTED will continue to receive RIP responses from the active gateways over the hyperchannel device.

Configuring a Point-to-Point Link

The OROUTED server can manage point-to-point links as long as there are RIP services at both ends of the links. If a host router at the other end of a link is not running a RIP service, then passive routing must be configured in the OROUTED gateways data set for the link. See “Configuring a Passive Route” on page 938.

Configuring a Default Route

A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the passive route definition in the gateways file or data set. For example, if the default destination router has a gateway address 9.67.112.1, then add the following entry to the data set:

```
net  0.0.0.0  gateway  9.67.112.1  metric  1  passive
```

Only one default route to a destination gateway or router can be specified. OROUTED currently does not support multiple default routes.

Configuring ORouteD with Enterprise Extender

If running ORouteD with the Enterprise Extender on a OS/390 TCP/IP stack connected to SNA via IUTSAMEH device, special configuration needs to be considered depending on the system environment. Assume that TOVTAM interface is the link name for the IUTSAMEH device and its home IP address is 9.2.1.1.

1. If there are no other OS/390 TCP/IP stacks in this MVS image, disable RIP on the TOVTAM interface to prevent ORouteD from sending any routing information over this interface. For example, in the ORouteD gateways file or data set:

   ```
   options interface TOVTAM 9.2.1.1 ripoff
   ```

2. If one or more OS/390 TCP/IP stacks exist in the MVS image, do the following:

- Define the link characteristics for the TOVTAM interface in the BSDROUTINGPARMS statement of TCP/IP configuration profile. In this definition, specify a subnet mask and a zero IP destination address.

  For example, assume that the subnet mask for the TOVTAM interface is 255.255.255.0:

  ```
  ; link  maxmtu       metric   subnet mask     dest_addr
  TOVTAM  DEFAULTSIZE    0     255.255.255.0       0
  ```

  If using RIP1 or RIP2B on this interface and depending on its subnet mask, ORouteD will use a subnet-directed broadcast address to send the routing information to other OS/390 TCP/IP stacks in this MVS image. If using RIP2 or RIP2M, ORouteD will use the RIP2 multicast address.

3. Repeat the above step for other OS/390 TCP/IP stacks running with ORouteD and configured with IUTSAMEH devices in this MVS image. Ensure that the RIP settings for the TOVTAM interfaces are identical so that the routing information exchanges will occur between OS/390 TCP/IP stacks running ORouteD in this MVS image.

Configuring OROUTED with VIPA

For more information on configuration options with VIPA, see the following topics:

- "Chapter 3. Virtual IP Addressing" on page 71
- "Configuring Static VIPAs for an OS/390 TCP/IP Stack" on page 74
- "Planning for Static VIPA Takeover and Takeback" on page 77
- "Configuring OROUTED to Split Traffic with VIPA"

Assume that you want to configure a VIPA address in one OS/390 TCP/IP stack as in Figure 46 on page 942. Following are the configuration steps:

1. Configure the DEVICE, LINK, HOME, ASSORTEDPARMS, and IPCONFIG statements for a VIPA address as described in "Configuring Static VIPAs for an OS/390 TCP/IP Stack" on page 74. The assigned VIPA address is 192.1.1.1 and link name is ″VIPA″.
2. Update the BSDROUTINGPARMS statement in *hlq*.PROFILE.TCPIP for the VIPA link, in addition to the physical links:

   ```
   BSDROUTINGPARMS true
       .
       .
     VIPA   DEFAULTSIZE   0  255.255.255.252 0
       .
       .
   ENDBSDROUTINGPARMS
   ```

   The subnet mask is associated with the VIPA link. The subnet mask must be assigned such that the VIPA interface is to be networked, subnetted, or supernetted. See the subnet mask parameter in "BSDROUTINGPARMS Statement" on page 140.

Configuring OROUTED to Split Traffic with VIPA

The purpose of splitting traffic is to reduce traffic load on network attachments by controlling the inbound and outbound traffic. The following techniques can be used to produce traffic splitting effects with fault tolerance benefit:

- Using Interface Metric and VIPA To Split Inbound/Outbound Traffic

In the multiple network attachments to the same network configuration, split inbound/outbound traffic can be achieved by configuring the metric on the primary interface to one higher than the secondary interface(s). From routing updates, an adjacent router uses the gateway of a secondary interface to reach the destination VIPA on the OS/390 server because the route to the gateway has a shorter metric. The primary interface is used for outbound traffic and a secondary interface is used for inbound traffic. The traffic splitting will function as long as the primary and at least one secondary interfaces are active. For information on configuring an interface metric, see the “BSDROUTINGPARMS Statement” on page 140. A VARY TCPIP,,CMD=OBEYFILE command for the BSDROUTINGPARMS statement can be used to update an interface metric for a link. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA” on page 941.

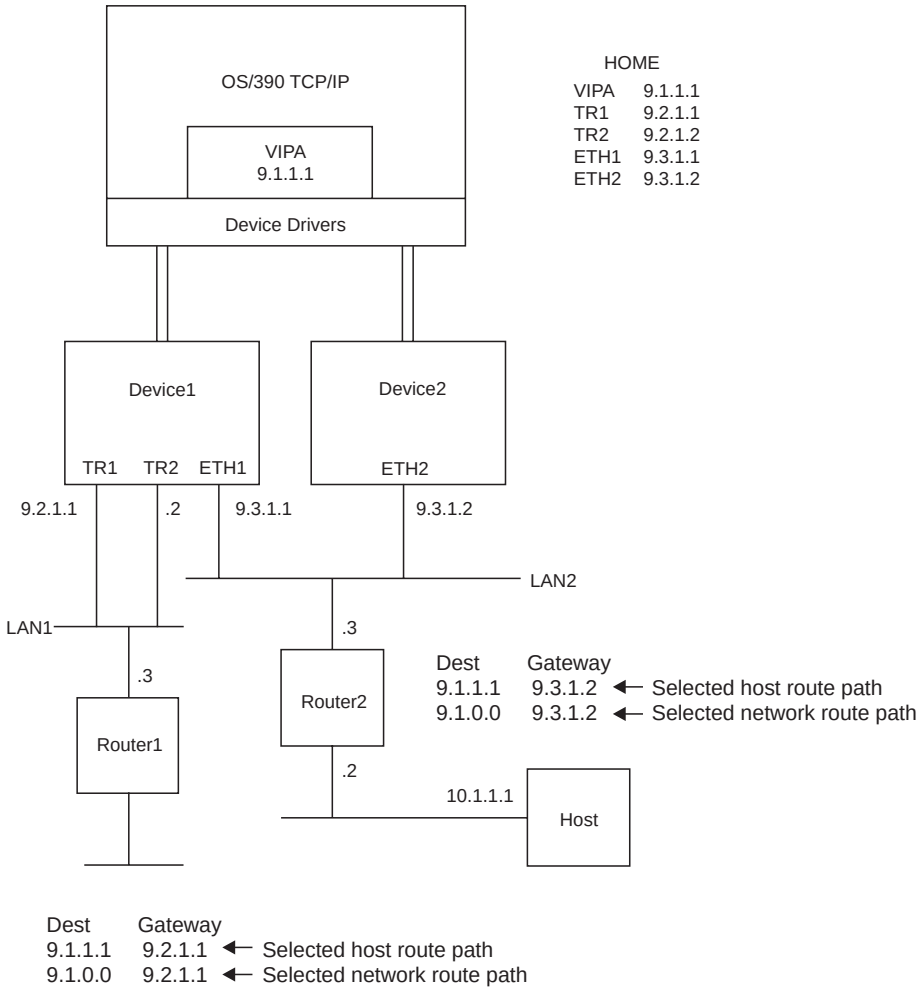


Figure 46. Single VIPA Configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

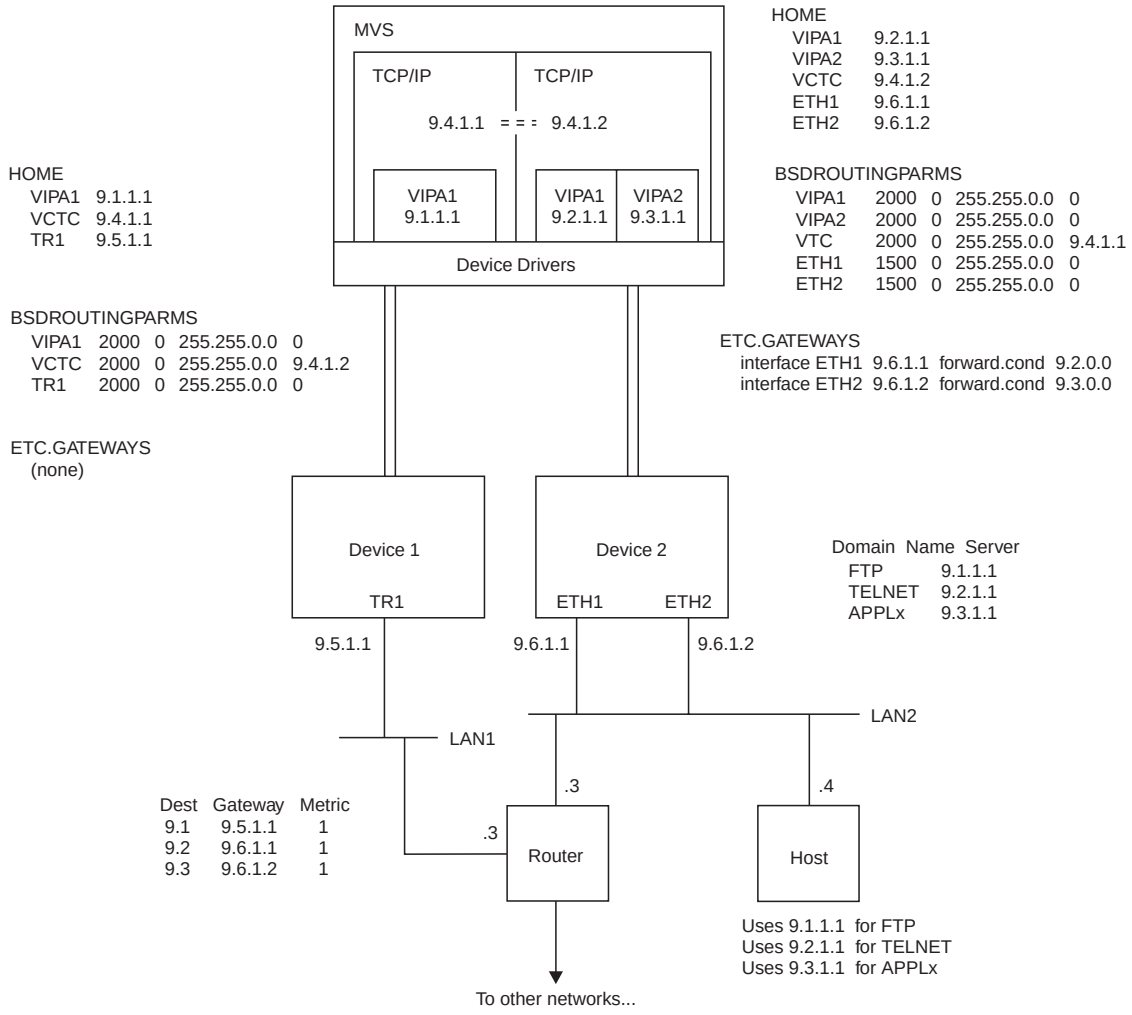


Figure 47. Multiple VIPA Configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

- Using Route Forwarding and VIPA to Split Session Traffic

  With multiple VIPAs in one TCP/IP stack, a VIPA can be assigned to a particular interface so that the VIPA can be reserved for session traffic (for example, FTP or TELNET). This is accomplished by using the route forwarding option in OROUTED. From routing updates, an adjacent router will have multiple gateways to reach the VIPAs on the OS/390 server. The adjacent router will use one gateway to reach one VIPA reserved for one type of session traffic and the other gateway to reach another VIPA reserved for another type of session traffic on the OS/390 server. For fault tolerance, it is recommended that the conditional option of route forwarding be used. For information on route forwarding, see the options statement in “Step 6: Configure the Gateways File or Data Set (Optional)” on page 928. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA” on page 941.

Configuring a Backup TCP/IP Stack with VIPA

To configure a backup OS/390 TCP/IP stack with a primary OS/390 TCP/IP stack’s VIPA address, do the following after a primary OS/390 TCP/IP stack is down:

- If OROUTED is running on the backup OS/390 TCP/IP stack, issue VARY TCPIP,,CMD=OBEYFILE commands to:
  1. Add new VIPA device and link.
  2. Add new HOME and BSDROUTINGPARMS statements for the new VIPA link. The HOME statement will have the primary OS/390 TCP/IP stack's VIPA address. Ensure that the HOME and BSDROUTINGPARMS statements are defined in one VARY TCPIP,,CMD=OBEYFILE command.
- If OROUTED is not running on the backup OS/390 TCP/IP stack, restart OROUTED and add the new VIPA device and link, HOME, and BSDROUTINGPARMS statements using the VARY TCPIP,,CMD=OBEYFILE command.

Restoring a Primary OS/390 TCP/IP Stack with VIPA

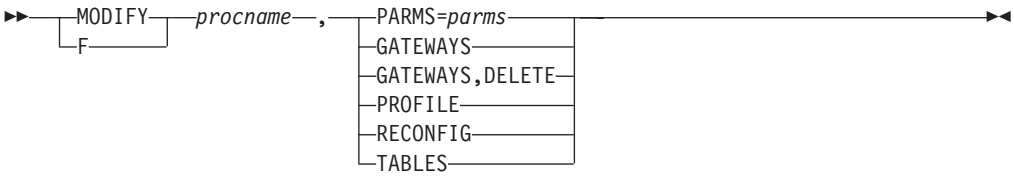
To restore a VIPA address to the primary OS/390 TCP/IP stack after it is no longer needed on the backup, do the following:

- If OROUTED is running on the backup OS/390 TCP/IP stack, issue a VARY TCPIP,,CMD=OBEYFILE command to add a new HOME statement with the primary OS/390 TCP/IP stack's VIPA address removed. Otherwise, issue the a VARY TCPIP,,CMD=OBEYFILE command and add a new HOME statement with the primary OS/390 TCP/IP stack's VIPA address removed and restart OROUTED. It is not necessary to add a new BSDROUTINGPARMS statement.
- Start OROUTED on the primary OS/390 TCP/IP stack after ensuring that the VIPA device and link, HOME, and BSDROUTINGPARMS statements are defined for the VIPA address that was temporarily reassigned to the backup OS/390 TCP/IP stack.

Controlling OROUTED with the MODIFY Command

You can control most of the OROUTED server functions from the operator's console using the MODIFY command. The following is the syntax and valid parameters.

Syntax



Parameters

procname

If OROUTED was started from a cataloged procedure, procname is the member name of that procedure. If OROUTED was started from the OS/390 shell, the procname is *userid*X, where *X* is the sequence number set by the system. To determine the sequence number, from the SDSF LOG window on TSO issue "/d omvs,u=*userid*". This will show the programs running under the userid *userid*. The procname can also be set using the environment variable _BPX_JOBNAME and then starting OROUTED in the shell background.

parms

Any one or more of the following separated by a space.

-d Enables printing of internal debug information to STDOUT.

-dp Trace packets to and from adjacent routers and received and sent RIP responses. Packets are displayed in data format. Output is to STDOUT.

-dq Disable all debug traces.

-f Flush all indirect routes known by OROUTED from IP routing tables.

-fh Flush all indirect host routes known by OROUTED from IP routing tables.

-g Enable default router broadcasting. When this option is specified, OROUTED will add a default route to its routing information and propagate it over all local interfaces.

-gq Disable default router.

-h Include host routes in addition to network-specific routes for the RIP dates. Adjacent routers must be able to receive host routes to prevent NETWORK UNREACHABLE problems from occurring.

-hq Disable supplying host routes (non-VIPA and VIPA).

-hv Include only VIPA host routes in addition to network-specific routes for the RIP responses. Adjacent routers must be able to receive host routes; otherwise, network or subnetwork portions of VIPA addresses must be unique for each OS/390 TCP/IP stack.

-hvq Disable supplying VIPA host routes.

-k Kill OROUTED. OROUTED will post a message to the console and to STDERR and then end.

-kdr Delete all dynamic RIP routes and kill OROUTED. OROUTED will post a message to the console and to STDERR and then end.

-s When OROUTED is started with *-q,* use *-s* to force supply of routing information.

-sd Enable supply default route.

-sdq Disable supply default route only.

-sdv Similar to svd parameter.

-sdvq Similar to svdq parameter.

-sl Enable supply local (directly-connected) routes only.

-slq Disables supply only local (directly-connected) routes.

-sq or -q
Disable supply all routes.

-st Supply triggered updates only

-stq Disable supply triggered updates

-sv Enable supply network-specific VIPA route.

-svq Disable supply network-specific VIPA route.

-svd (or sdv)
Enable supply network-specific VIPA and default routes.

-svdq (or sdvq)
Disable supply network-specific VIPA and default routes.

-svh (or shv)
Enable supply VIPA (network-specific and host) routes only.

-svhq (or shvq)
Disables supplying VIPA routes (network-specific and host).

-t Enable or disable traces. Up to 4 -t parms are allowed.

-tq Disable all traces.

See Table 43 on page 936 for more information.

GATEWAYS
Reread the GATEWAYS file or data set.

GATEWAYS,DELETE
Reread the GATEWAYS file or data set and delete all routes listed.

PROFILE
Reread the OROUTED profile file or data set.

RECONFIG
Performs dynamic reconfiguration of interfaces and local routes when the interface parameters have changed after a VARY TCPIP,,CMD=OBEYFILE command has been issued for a data set containing modified HOME and BSDROUTINGPARMS statements. Issue this command for OROUTED after a VARY TCPIP,,CMD=OBEYFILE command has been issued to reflect dynamic changes made.

TABLES
Display RIP routing and interface tables (internal to OROUTED).

Examples

The following commands would pass parameters to an OROUTED server started with a procedure named OROUTED.

```
MODIFY OROUTED,PARMS=-t -t -s
F OROUTED,GATEWAYS,DELETE,PARMS=-sq
F OROUTED,PARMS=-h,PROFILE,GATEWAYS
```

Chapter 26. Configuring OMPROUTE

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the OMPROUTE application. It explains OMPROUTE’s use of the Open Shortest Path First (OSPF) protocol and the Routing Information Protocol (RIP). This information will help you decide if this application is suitable for your network and, if so, whether the OSPF or RIP protocol (or both) is best suited for you.

Understanding OMPROUTE

OMPROUTE 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). It provides an alternative to the static TCP/IP gateway definitions. When configured properly, the MVS host running with OMPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be used to dynamically maintain the host routing table. For example, OMPROUTE can detect when a route is created, is temporarily unavailable, or 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.

OMPROUTE has the following characteristics:

- It is an OS/390 UNIX application. It requires the Hierarchical File System (HFS) to operate.
- OMPROUTE can be started from a MVS started procedure, from the OS/390 shell, or from AUTOLOG.
- OMPROUTE needs to be started by a RACF-authorized user ID.
- OMPROUTE needs to be in an APF authorized library.
- A one-to-one relationship exists between an instance of OMPROUTE and a stack. OSPF/RIP support on multiple stacks, via the split-stack function, would require multiple instances of OMPROUTE.
- OMPROUTE and RouteD cannot run on the same stack concurrently.
- All dynamic routes are deleted from the routing table upon initialization of OMPROUTE.
- ICMP Redirects are ignored when OMPROUTE is active.
- Unlike RouteD, OMPROUTE does not make use of the BSD Routing Parameters. Instead, the Maximum Transmission Unit (MTU), subnet mask, and destination address parameters are configured via the OSPF_Interface, RIP_Interface, and Interface statements in the OMPROUTE configuration file.
- When OMPROUTE is configured for both the OSPF and RIP protocols, routes that are learned through the OSPF protocol take precedence over routes that are learned through the RIP protocol.
- OMPROUTE uses the MVS operators console, SYSLOGD, CTRACE, and STDOUT for its logging and tracing. The MVS operators console and SYSLOGD are used for major events such as initialization, termination, and error conditions. CTRACE is used for tracing the receipt and transmission of OSPF/RIP packets

as well as communications between OMPROUTE and the TCP/IP stack. STDOUT is used for detailed tracing and debugging.

- An Interface must be configured to OMPROUTE via the OSPF_Interface or RIP_Interface configuration statement if it is desired that the specified routing protocol be communicated over it.
- Interfaces which are not to be involved in the communication of the RIP or OSPF protocol (such as VIPA interfaces) must be configured to OMPROUTE via the Interface configuration statement (unless it is a non-point-to-point interface and all default values specified on the Interface statement are acceptable).
- OMPROUTE uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.
- OMPROUTE is enhanced with Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The VIPA routes are included in the 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 MVS host.
- OMPROUTE allows for the generation of multiple, equal-cost routes to a destination, thus providing load splitting support. For OSPF and RIP, up to four equal-cost multipath routes are allowed. However, if you are using RIP, only up to four directly-connected network routes are allowed.
- The OMPROUTE subagent implements RFC 1850 for the OSPF (Open Shortest Path First) Protocol.
- Unlike OROUTED, OMPROUTE does not use the environment variable GATEWAYS_FILE. If static routes are to be used together with OMPROUTE (this is not recommended), they must be configured to the TCP/IP stack using the GATEWAY statement in the TCP/IP profile. OMPROUTE will learn of these static routes through communications with the TCP/IP stack.

Notes:

1. Use of static routes (defined via the GATEWAY TCP/IP configuration statement) along with OMPROUTE is not recommended. These static routes may interfere with the discovery of a better route to the destination as well as inhibit the ability to switch to another route if the destination should become unreachable via the static route.
2. If static routes must be defined, all static routes will be considered to be of equal cost and static routes will not be replaced by OSPF or RIP routes. Use extreme care when working with static routes and OMPROUTE. Set IMPORT_STATIC_ROUTES = YES on the AS_Boundary Routing configuration statement or set SEND_STATIC_ROUTES = YES on the RIP_Interface configuration statement if you wish for the static routes to be advertised to other routers.
3. For details on the statements in the OMPROUTE configuration file, see “OMPROUTE Configuration File” on page 960.
4. Display of OMPROUTE information is performed using the DISPLAY command. For details on OMPROUTE’s DISPLAY commands, see “Using OMPROUTE DISPLAY Commands” on page 993.
5. Modification of OMPROUTE information is performed using the MODIFY command. For details on OMPROUTE’s MODIFY commands, see “Controlling OMPROUTE with the MODIFY Command” on page 958.
6. If OMPROUTE will be communicating through the OSPF or RIPv2 protocol over a token ring media, and there will be routers attached to that token ring that are

not listening (at the DLC layer) for the token ring multicast address 0xC000.0004.0000, the following TRANSLATE statement will be required in the TCP profile:

```
TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFFFF linkname
```

Without this statement, OSPF and RIPv2 multicast packets will be discarded at the DLC layer by those routers that are not listening for the token ring multicast address.

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 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.

Note: Before using the OSPF function in OMPROUTE, read RCF 1583 to understand the OSPF routing function and how to use it to manage your network.

OSPF provides support for equal-cost multipath, provides for the authentication of routing updates, and utilizes IP multicast when sending/receiving the updates. An area routing capability is provided, enabling an additional level of routing protection and a reduction in routing protocol traffic.

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.

In a link-state routing protocol, each router maintains a database describing the Autonomous System's topology. Each participating router has an identical database. 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.

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 appears on the tree as leaves. When several equal-cost routes to a destination exist, traffic is distributed equally among them. The cost of a route is described by a single dimensionless metric.

OSPF 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 information hiding 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.

OSPF 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).

OSPF 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 area. This enables some areas to use authentication while others do not.

Externally derived routing data (for example, routes learned from the Routing Information Protocol [RIP]) 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, enabling the passing of additional information between routers on the boundaries of the Autonomous System.

Routing Information Protocol (RIP)

RIP is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. IGPs are used to manage the routing information of a single autonomous system, or a single piece of the TCP/IP network. RIP is based on the Bellman-Ford or the distance-vector algorithm. RIP has many limitations and is not suited for every TCP/IP environment. Before using the RIP function in OMPROUTE, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. See “Appendix D. Related Protocol Specifications (RFCs)” on page 1177 for more information about RFCs 1058 and 1723.

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. A gateway is defined as zero hops from its directly connected networks, one hop from networks that can be reached through one gateway, and so on. 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 periodically 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 and sets all routes through that router to a metric of 16 (infinity). 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 to provide EGP-RIP and BGP-RIP interactions

  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. OMPROUTE will not generate route tags, but will preserve them in received routes and readvertise 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 being included in the routing information to eliminate packets being routed through extra hops in the network. OMPROUTE will not generate immediate next hops, but will preserve them if they are included in the RIP packets.
- Multicasting to reduce load on hosts

  An 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.

The Configuration Process

The steps to configure OMPROUTE are as follows and are described below:

1. Create the OMPROUTE configuration file
2. Reserve the RIP UDP port (if using the RIP protocol)
3. Update the resolver configuration file
4. Update the OMPROUTE cataloged procedure (optional)
5. Specify the RIP UDP port number in the SERVICES file or data set (if using the RIP protocol)
6. RACF authorize user IDs for starting OMPROUTE
7. Start syslogd
8. Update the OMPROUTE environment variables (optional)
9. Create static routes (optional)

Step 1: Create the OMPROUTE Configuration File

The following is the search order used by OMPROUTE to locate the configuration data set or file:

1. If the environment variable OMPROUTE_FILE has been defined, OMPROUTE uses the value as the name of an MVS data set or HFS file to access the configuration data. The syntax for a MVS data set name is ″//'mvs.dataset.name'″. The syntax for a HFS file name is ″/dir/subdir/file.name″.
2. /etc/omproute.conf
3. *hlq*.ETC.OMPROUTE.CONF

A sample configuration file is provided as member EZAORCFG in data set *hlq.SEZAINST*. For a description of the syntax rules for the OMPROUTE configuration file as well as details on each of the configuration statements, refer to

"OMPROUTE Configuration File" on page 960. For a description of how the *hlq* value is determined, see "Step 3: Update the Resolver Configuration File".

Step 2: Reserve the RIP UDP Port (If Using the RIP Protocol)

If the RIP protocol of OMPROUTE is going to be used, UDP port 520 should be reserved for OMPROUTE. This is done by adding the name of the member containing the OMPROUTE cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  520 UDP OMPROUTE
```

If you want to be able to start omproute from the OS/390 shell, use the special name OMVS as follows:

```
PORT
  520 UDP OMVS
```

This enables the entire ″OMVS job group″ (that is, all OS/390 shell users). Only RACF-authorized users can start omproute.

Note: As discussed in "AUTOLOG Statement" on page 137, if a procedure 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, TCPIP attempts to cancel that procedure and start it again. Such a procedure, if placed in the AUTOLOG list, would be subject to being periodically cancelled and restarted by TCP/IP because it does not have a permanent listening connection established.

Therefore, if OMPROUTE is being started with AUTOLOG and only the OSPF protocol is being used (no RIP protocol and, therefore, no listening connection on the RIP UDP port), it is important that one of the following be done:

- Ensure that the RIP UDP port (520) is not reserved by the PORT statement in the TCP/IP profile.
- Add the NOAUTOLOG parameter to the PORT statement in the TCP/IP profile. For example,

  ```
  PORT
    520 UDP OMPROUTE NOAUTOLOG
  ```

Failure to take one of the above actions will result in OMPROUTE being periodically cancelled and restarted by TCP/IP.

Step 3: Update the Resolver Configuration File

The resolver configuration file or data set contains keywords (DATASETPREFIX and TCPIPjobname) that are used by OMPROUTE. The value assigned to DATASETPREFIX will determine the high-level qualifier (hlq). The hlq is used in the search order for the OMPROUTE configuration file. If no DATASETPREFIX keyword is found, a default of TCPIP is used. The value assigned to TCPIPjobname will be used as the name of the TCP/IP stack with which OMPROUTE establishes a connection.

For a description of the search order used by the resolver to locate the resolver configuration data set or file, see "Search Order and Configuration Files for TCP/IP Applications" on page 19.

Step 4: Update the OMPROUTE Cataloged Procedure (Optional)

If OMPROUTE is to be started by a procedure, update the cataloged procedure OMPROUTE by copying the sample in hlq.SEZAINST(OMPROUTE) to your system or recognized PROCLIB. Specify OMPROUTE parameters and change the data set names to suit your local configuration.

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZBORPRC
//*
//*        5647-A01 (C) Copyright IBM Corp. 1998.
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//OMPROUTE EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/omproute '
//*
//*        Provide environment variables to run with the
//*        desired stack and configuration.  As an example,
//*        the file specified by STDENV could have these
//*        three lines in it:
//*
//*           export RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*           export OMPROUTE_FILE=/u/usernnn/config.tcpcs2
//*           export OMPROUTE_DEBUG_FILE=/tmp/logs/omproute.debug
//*
//*        For information on the above environment variables,
//*        refer to the IP CONFIGURATION GUIDE.
//*
//STDENV   DD PATH='/u/usernnn/envcs2',
//         PATHOPTS=(ORDONLY)
//*
//*        To save the program output to standard out to
//*        an HFS file, define STDOUT as show below:
//*
//STDOUT   DD PATH='/tmp/omproute.stdout',
//         PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        To capture standard error messages in a file, define
//*        STDERR
//*
//STDERR   DD PATH='/tmp/omproute.stderr',
//         PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Step 5: Specify the RIP UDP Port Number in the SERVICES File or Data Set (If Using the RIP Protocol)

The services data set or file contains the relationship between services and port numbers. The portion of the services file relevant to OMPROUTE is shown in Figure 48 on page 956. The data set or file must exist for the RIP protocol of OMPROUTE to operate.

```
route           520/udp         router routed
```

Figure 48. Portion of the Services File Relevant to OMPROUTE

For a description of the search order used to locate the services data set or file, see “Search Order and Configuration Files for TCP/IP Applications” on page 19.

Step 6: RACF-Authorize User IDs for Starting OMPROUTE

(and thus reduce risk of an unauthorized user starting it and affecting the contents of the routing table), the appropriate users must be RACF authorized to the entity MVS.ROUTEMGR.OMPROUTE. To do this, the following commands must be entered from a RACF user ID, substituting the authorized user ID on the ID(*userid*) parameter:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Step 7: Start syslogd

To get only the urgent OMPROUTE messages written to the MVS console, you should have syslogd running while OMPROUTE is running. Syslogd sends the non-urgent messages to the HFS message log.

Step 8: Update the OMPROUTE Environment Variables (Optional)

The following environment variables are used by OMPROUTE and can be tailored to a particular installation:

- RESOLVER_CONFIG

  The RESOLVER_CONFIG variable is used by OMPROUTE to locate the resolver configuration file. For more information on OMPROUTE’s use of the resolver configuration file, see “Step 3: Update the Resolver Configuration File” on page 954, and “Search Order and Configuration Files for TCP/IP Applications” on page 19. For more information about the RESOLVER_CONFIG environment variable, refer to *OS/390 UNIX System Services Planning*.
- OMPROUTE_FILE

  The OMPROUTE_FILE variable is used by OMPROUTE in the search order for the OMPROUTE configuration file. For details on the search order used for locating this configuration file, see “Step 1: Create the OMPROUTE Configuration File” on page 953.
- OMPROUTE_DEBUG_FILE

  The OMPROUTE_DEBUG_FILE variable is used by OMPROUTE to override the debug output destination. For more information on using this environment variable, see “OMPROUTE Parameters” on page 958.

Step 9: Create Static Routes (Optional)

OMPROUTE does not use the environment variable GATEWAYS_FILE to initialize static routes. To create static routes, use the TCP/IP profile statement GATEWAY. For information on the syntax of the statement, see “GATEWAY Statement” on page 193.

During intialization, OMPROUTE sets up static routes by reading the internal gateways table set up by TCP/IP when it intialized.

Starting OMPROUTE

After the necessary RACF authorization has been defined, OMPROUTE can be started from an MVS procedure, from the OS/390 shell, or from AUTOLOG.

- You can start OMPROUTE from the MVS operators console by starting the OMPROUTE start procedure. A sample start procedure is provided with the product in *hlq*.SEZAINST(OMPROUTE).
- You can start omproute from the OS/390 shell by starting OMVS and then issuing the omproute command and, optionally, any parameters. For information on parameters, see “OMPROUTE Parameters” on page 958.
- You can use the AUTOLOG statement to start OMPROUTE automatically during TCP/IP initialization. Insert the name of the OMPROUTE start procedure in the AUTOLOG statement of the hlq.PROFILE.TCPIP data set.

```
AUTOLOG
  OMPROUTE
ENDAUTOLOG
```

Note: As discussed in “AUTOLOG Statement” on page 137, if a procedure 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, TCPIP attempts to cancel that procedure and start it again. Such a procedure, if placed in the AUTOLOG list, would be subject to being periodically cancelled and restarted by TCP/IP because it does not have a permanent listening connection established.

Therefore, if OMPROUTE is being started with AUTOLOG and only the OSPF protocol is being used (no RIP protocol and, therefore, no listening connection on the RIP UDP port), it is important that one of the following be done:

– Ensure that the RIP UDP port (520) is not reserved by the PORT statement in the TCP/IP profile.

– Add the NOAUTOLOG parameter to the PORT statement in the TCP/IP profile. For example,

```
PORT
  520 UDP OMPROUTE NOAUTOLOG
```

Failure to take one of the above actions will result in OMPROUTE being periodically cancelled and restarted by TCP/IP.

In a Common INET environment, OMPROUTE will attempt to connect to a stack whose name is determined by the TCPIPjobname keyword from the resolver configuration data set or file. In configurations with multiple stacks, a copy of OMPROUTE must be started for each stack that requires OMPROUTE services. To associate OMPROUTE with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique TCPIPjobname.

When running from a MVS procedure, the environment variables can be set by using the STDENV DD statement in the OMPROUTE procedure. For information concerning the environment variables used by OMPROUTE, see “Step 8: Update the OMPROUTE Environment Variables (Optional)” on page 956.

OMPROUTE Parameters

OMPROUTE accepts the -tn command line parameter. This parameter is valid when starting the program from an MVS procedure, from the OS/390 shell, or from AUTOLOG.

The -tn option specifies the external tracing level, where *n* is a supported trace level. It is intended for customers, testers, service, or developers, and provides information on the operation of the routing application. This option can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of testcases, and so on. The following levels are supported:

```
1 = Informational messages
2 = Formatted packet trace
```

These option levels are cumulative—level 2 includes level 1. For example, -t2 provides formatted packet trace and informational messages. Output from this option is written to stdout with one exception. The exception is when the routing application was started with no tracing, and then a MODIFY command is issued to enable tracing. In this case, the output destination defaults to the file omproute_debug in the current temporary directory (the default is /tmp). The debug output destination will have an override capability via the use of an environment variable (OMPROUTE_DEBUG_FILE).

If OMPROUTE is to be started from an MVS procedure, add your parameters to PARM='PGM in the OMPROUTE cataloged procedure. For example: //PARM='PGM /usr/sbin/omproute -t1'.

If omproute is to be started from an OS/390 shell command line, enter the parameters on the command line.

For either method of starting OMPROUTE, parameters can be specified in mixed case.

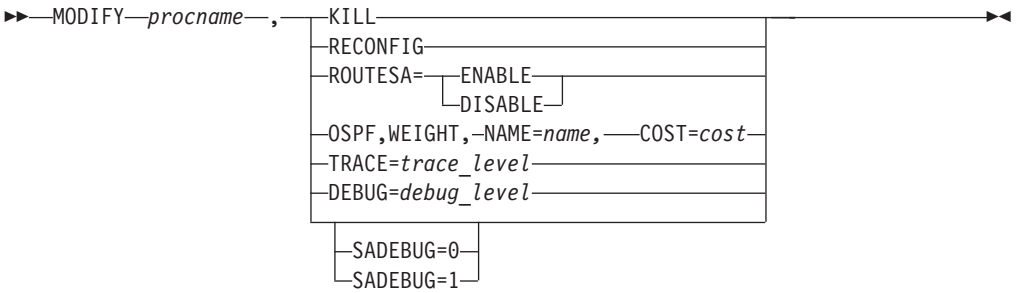
Note: Use of the -tn parameter affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

Controlling OMPROUTE with the MODIFY Command

You can control OMPROUTE from the operator's console using the MODIFY command. MODIFY commands are available to perform the following functions:

- "Stopping OMPROUTE" on page 959
- "Rereading the Configuration File" on page 959
- "Enabling or Disabling the OMPROUTE Subagent" on page 959
- "Changing the Cost of OSPF Links" on page 960
- "Controlling OMPROUTE Tracing" on page 960

The syntax for this command is:



Stopping OMPROUTE

OMPROUTE can be stopped in several ways:

- From MVS, issue *P <procname>* or *MODIFY <procname>,KILL*.

  If OMPROUTE was started from a cataloged procedure, procname is the member name of that procedure. If OMPROUTE was started from the OS/390 shell, procname is *userid*X, where X is the sequence number set by the system. To determine the sequence number, from the SDSF LOG window on TSO, issue "/d omvs,u=*userid*". This will show the programs running under the userid *userid*. The procname can also be set using the environment variable _BPX_JOBNAME and then starting OMPROUTE in the shell background.
- From an OS/390 shell superuser ID, issue the kill command to the process ID (PID) associated with OMPROUTE. To find the PID, use one of the following methods:
  - From the MVS console, issue *D OMVS,U=userid*, or issue */D OMVS,U=userid* at the SDSF LOG window on TSO (where *userid* is the user ID that started omproute from the shell).
  - Issue the ps -ef command from the OS/390 shell.
  - Write down the PID when you start OMPROUTE.

For information on the environment variable _BPX_JOBNAME, refer to *OS/390 UNIX System Services Planning*. For information on the *D OMVS,U=userid* command, see *OS/390 MVS System Commands*.

Rereading the Configuration File

The MODIFY *<procname>*,RECONFIG command is used to reread the OMPROUTE configuration file. This command ignores all statements in the configuration file except new OSPF_Interface, RIP_Interface, and Interface statements. These new configuration statements must be re-read from the configuration file through this command prior to the interface being configured to the TCP/IP stack.

Enabling or Disabling the OMPROUTE Subagent

Use the MODIFY *procname*,ROUTESA=ENABLE command or the MODIFY *procname*,ROUTESA=DISABLE command to enable or disable the OMPROUTE subagent.

Note: To change any other value on the ROUTESA_CONFIG statement, the OMPROUTE application must be recycled.

The OMPROUTE subagent implements RFC 1850 for the OSPF (Open Shortest Path First) Protocol. The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, see “ROUTESA_CONFIG Statement” on page 991.

Changing the Cost of OSPF Links

The cost of an OSPF interface can be dynamically changed using the MODIFY *procname*,OSPF,WEIGHT,NAME=*name*,COST=*cost* command. 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 the router is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF interface in the configuration file.

Controlling OMPROUTE Tracing

The MODIFY *procname*,TRACE=*trace_level* command is used to start, stop, or change the level of OMPROUTE tracing. The different trace levels available and their descriptions are as follows:

TRACE=0
Turns off OMPROUTE tracing.

TRACE=1
Gives all the informational messages.

TRACE=2
Gives the informational messages plus formatted packet tracing.

Note: Use of OMPROUTE tracing affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

OMPROUTE Configuration File

Statements in the OMPROUTE configuration file have the following syntax:

```
type tag=value tag=value.. .. ..;
where:
type         Specifies what is to be configured
tag=value    Specifies a parameter and its associated value.
type=value   Used for statements that have only a single parameter.
```

The following are the syntax rules for the OMPROUTE configuration statements:

- Types, tags, and values can be specified in mixed case.
- Every configuration statement must terminate 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.

OSPF Configuration Statements

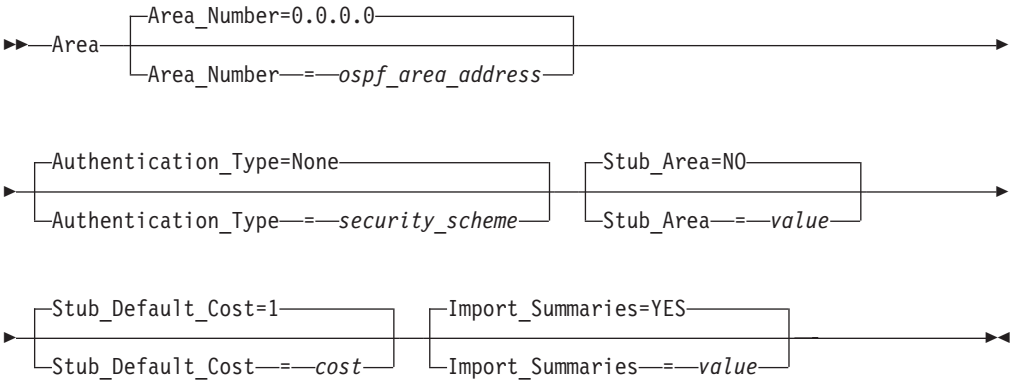
The following sections describe the OSPF configuration statements.

Note: Statements with only optional operands must have at least one operand coded, even if all operands have defaults.

AREA

Sets the parameters for an OSPF area. If no areas are defined, the router software assumes that all the router's directly attached networks belong to the backbone area (area ID 0.0.0.0).

Syntax:



Parameters:

Area_Number
The OSPF area address in dotted decimal.

Authentication_Type
The security scheme to be used in the area. Valid values for authentication types are ″Password″, which indicates a simple password; or ″None″, which indicates that no authentication is necessary to pass packets.

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.

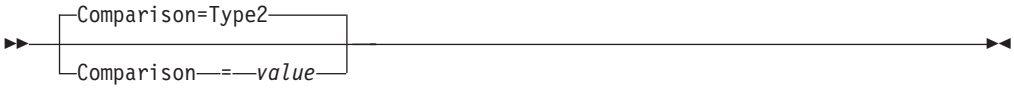
Stub_Default_Cost
The cost OSPF associates with the default route to it's border area router. Valid values are 1 to 65535.

Import_Summaries
Determines whether this stub area will import a routing summary from a neighbor area. Valid values are YES or NO.

COMPARISON

Tells the router where external routes fit in the 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 AS. 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.

Syntax:



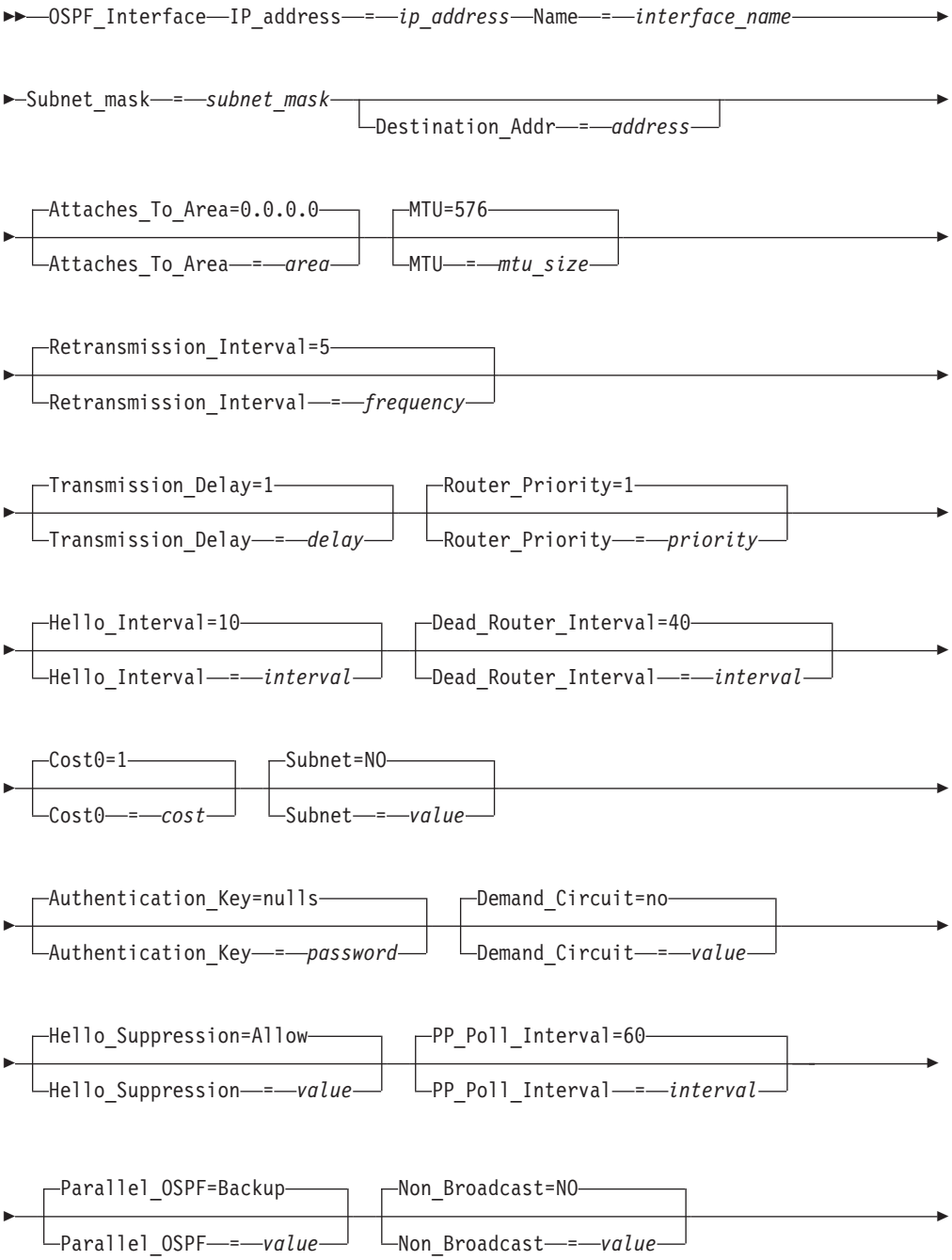
Parameters:

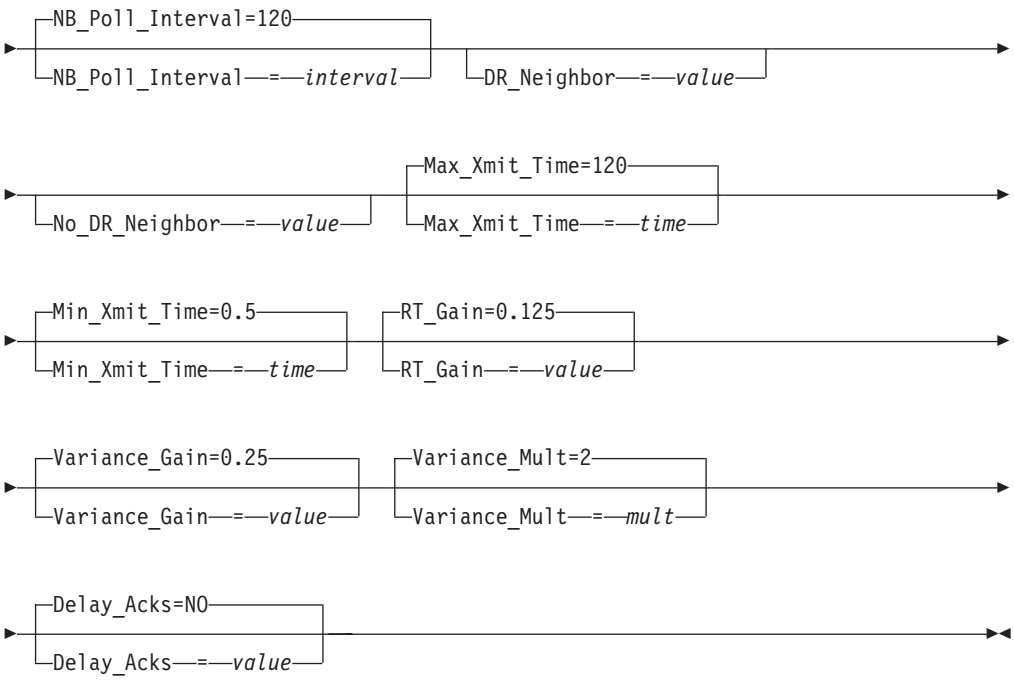
Comparison
Compare to type 1 or 2 externals. Valid values are Type1 (or 1) or Type2 (or 2).

OSPF_INTERFACE

Sets the OSPF parameters for the router's network interfaces. This statement will need to be replicated in the config file for each IP interface over which OSPF will operate.

Syntax:





Parameters:

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 wild cards. The valid wild-card specifications are below. The result of coding a wild card value is that all configured interfaces whose IP address matches the wild card will be configured as OSPF interfaces. Configured interface IP addresses will be matched against possible wild cards in the order they appear below with x.y.z.* being the best match, x.y.*.* being 2nd best, and so forth.

```
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.*  - Same as ALL
ALL      - Same as *.*.*.*
```

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 profile. Valid values are any 16 characters. This parameter is ignored on wildcard interface definitions.

Subnet_Mask
The subnet mask of the subnet this interface attaches to. This value must be the same for all routers attached to a common network.

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 OMPROUTE 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
The maximum transmission unit size for OSPF to add to the routing table for routes that take this interface. Valid values are 0 to 65535.

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 to a particular interface, it is aged by this configured transmission delay. Valid values are 1 to 65535 seconds.

Router_Priority
This value is used for broadcast and non-broadcast multiaccess networks to elect the designated router, with the highest priority router being elected. For point-to-point links, this value should be 0, which means that this router must not be elected the designated router for its network. Valid values are 0 to 255.

Hello_Interval
This parameter defines the number of seconds between OSPF Hello packets being sent out this interface. This value must be the same for all routers attached to a common network. Valid values are 1 to 255 seconds.

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 cost for this interface. The cost is used to determine the shortest path to a destination. Valid values are 1 to 65535. For information on configuring the cost of OSPF interfaces, see “Costs for OSPF Links” on page 1027.

Subnet
For an interface to a point-to-point serial line, this option enables the advertisement of a stub route to the subnet that represents the serial line rather than the host route for the other router’s address. Valid values are YES or NO.

Authentication_Key
The password for OSPF routers attached to this subnet. Coded when Authentication_Type=Password on the AREA statement for the virtual link’s transit area. This value must be the same for all routers attached to a common

network. Valid values are any characters from code page 1047 up to 8 characters in length coded within double quotes or any hex string which begins with 0x.

Note: If the value is entered in characters (rather than the hex string), that value is case sensitive.

Demand_Circuit
This parameter causes Link State Advertisements (LSAs) flooded over this interface to not age out. This is required if they will not be refreshed over this interface. Only LSA's with real changes will be advertised. Valid values are YES or NO.

Hello_Suppression
This parameter is meaningful only if Demand_Circuit is coded YES. This parameter allows you to configure the interface to request Hello suppression. This parameter is useful for point-to-point and point-to-multipoint interfaces. 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 poll interval (in seconds) to be used by OSPF to try to reestablish a connection when the point-to-point line is down because there was a failure to transmit data but the interface is still available. This parameter is meaningful only if Demand_Circuit is coded YES and Hello_Supression has been enabled. Valid values are 0-65535.

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 OMPROUTE and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If none of the interfaces to the common subnet are configured as primary, a primary interface will be selected by OMPROUTE. Valid values are ″Backup″ and ″Primary″.

Non_Broadcast
If the router is connected to a non-broadcast, multi-access network (NBMA), such as X.25, Frame Relay, Hyperchannel, or ATM networks, coding a Non_Broadcast helps the router discover its neighbors. This configuration is only necessary if the router will be eligible to become the designated router of the NBMA network. 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, and 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 non-broadcast multi-access and point-to-multipoint 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.

No_DR_Neighbor
Configures non-designated router eligible neighbors adjacent to the router over this interface. In non-broadcast multi-access and point-to-multipoint 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.

Max_Xmit_Time
The maximum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 999.990.

Min_Xmit_Time
The minimum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

RT_Gain
The round trip gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

Variance_Gain
The variance gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

Variance_Mult
The variance multiplier value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

Delay_Acks
The delay acknowledgments value to add to the routing table for routes that take this interface. Specifying YES delays transmission of acknowledgements when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgements being returned immediately. Valid values are YES and NO.

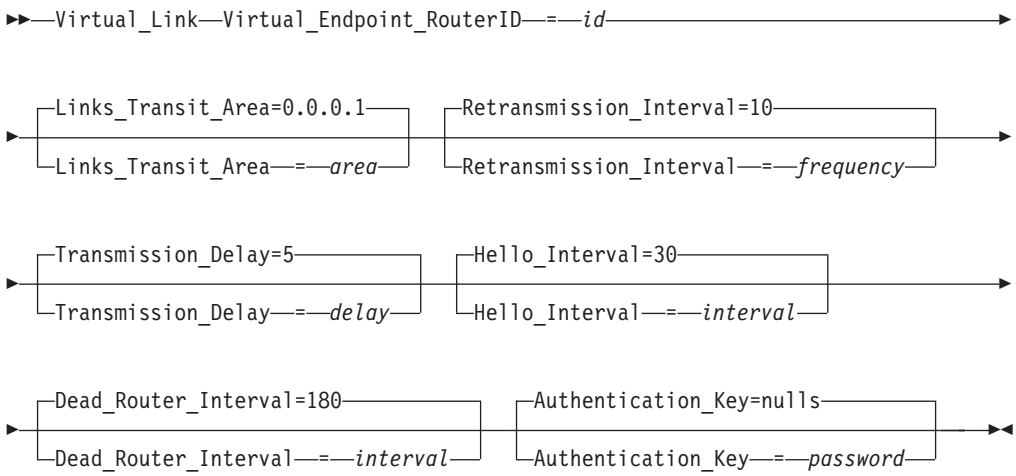
VIRTUAL_LINK

Configures virtual links between any 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 non-backbone area. Virtual links are used to maintain backbone connectivity and must be configured at both endpoints.

Note: OSPF virtual links are not to be confused with Virtual IP Address support (VIPA).

Syntax:



Parameters:

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 non-backbone, non-stub 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 non-backbone and non-stub 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 interface. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent to a particular interface, it is 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 this interface. 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.

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 password for OSPF routers attached to this subnet. Coded when Authentication_Type=Password on the AREA statement for the area where this interface attaches. This value must be the same for all routers attached to a common network. Valid values are any characters from code page 1047 up to 8 characters in length coded within double quotes or any hex string which begins with 0x.

Note: If the value is entered in characters (rather than the hex string), that value is case sensitive.

ROUTERID

Every router in an 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 the RouterID statement is specified, the value configured is used as an OSPF router ID. This value must be one of the stack's configured interface IP addresses.

  Note: 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.

Syntax:

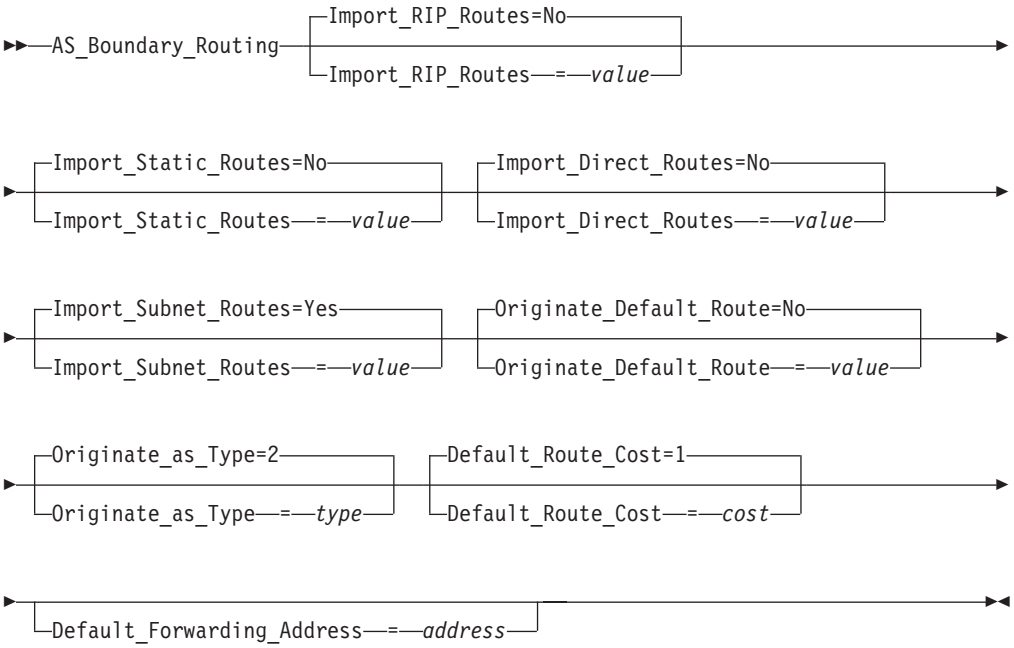
Parameters:

RouterID
RouterID as previously described.

AS_BOUNDARY_ROUTING

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 with their cost equal to their routing table cost. They are all 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.

Syntax:



Parameters:

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 OSPF should advertise this router as default router.

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 16777215.

Default_Forwarding_Address
Specifies the forwarding address that will be used in the imported default route.

RANGE

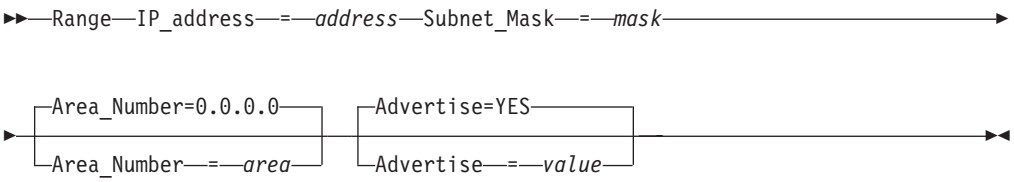
Adds ranges to OSPF areas. OSPF areas can be defined in terms of address ranges. External to the area, a single route is advertised for each address range. For example, if an OSPF area were to consist of all subnets of the class B network 128.185.0.0, it would be defined as consisting of a single address 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. There are two choices:

- When OSPF is configured to advertise the range, a single inter-area route is advertised for the range if at least one component route of the range is active within the area.
- When OSPF is configured not to advertise the range, no inter-area 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, OSPF will not function correctly if the area is partitioned but is connected by the backbone.

Syntax:



Parameters:

IP_Address
Common subnet portion of IP addresses in this range. Valid values are valid (sub)network addresses.

Subnet_Mask
Subnet mask with respect to the network range defined in IP_Address.

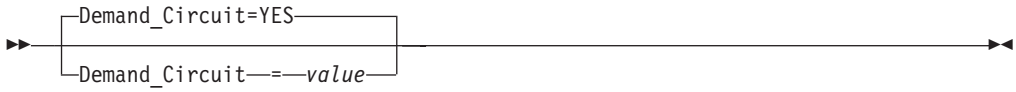
Area_Number
Area number for which to add this range. Valid values are any defined areas.

Advertise
Specifies whether this range will be advertised to other areas. Valid values are YES or NO.

DEMAND_CIRCUIT

Global demand circuit statement. Coding YES enables demand circuits. Demand circuit parameters can then be coded on the OSPF_Interface statement.

Syntax:



Parameters:

Demand_Circuit
Valid values are YES or NO.

RIP Configuration Statements

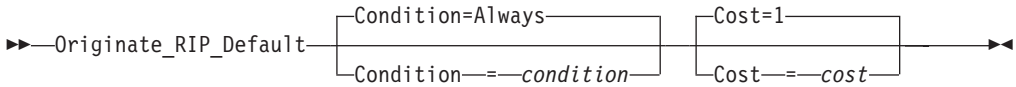
This section contains descriptions of the RIP configuration statements:

- ORIGINTE_RIP_DEFAULT
- RIP_INTERFACE
- ACCEPT_RIP_ROUTE
- FILTER
- Send_Only

ORIGINATE_RIP_DEFAULT

Indicates under what conditions RIP will support Default route (destination/mask 0.0.0.0/0.0.0.0) generation.

Syntax:



Parameters:

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.

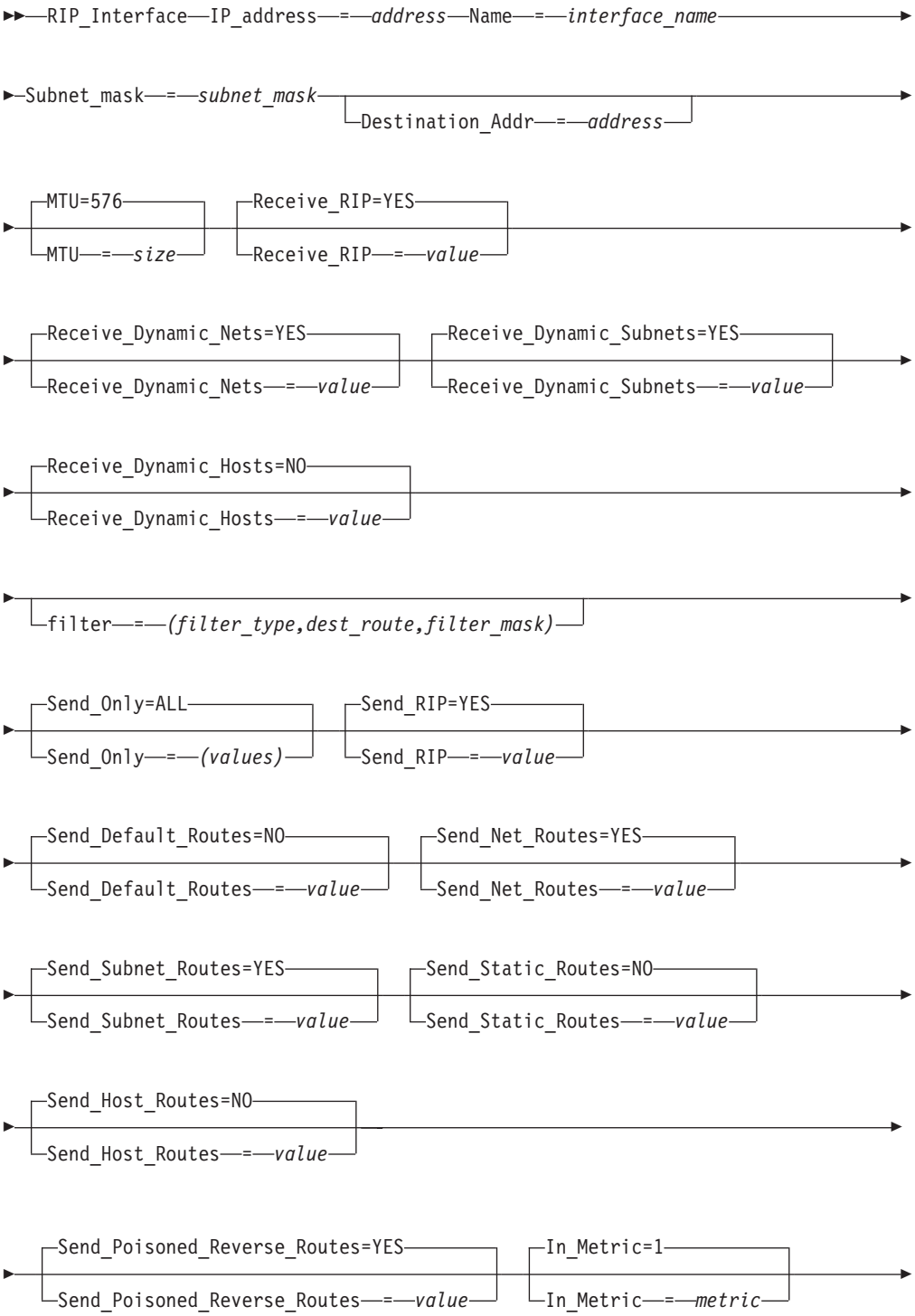
Cost

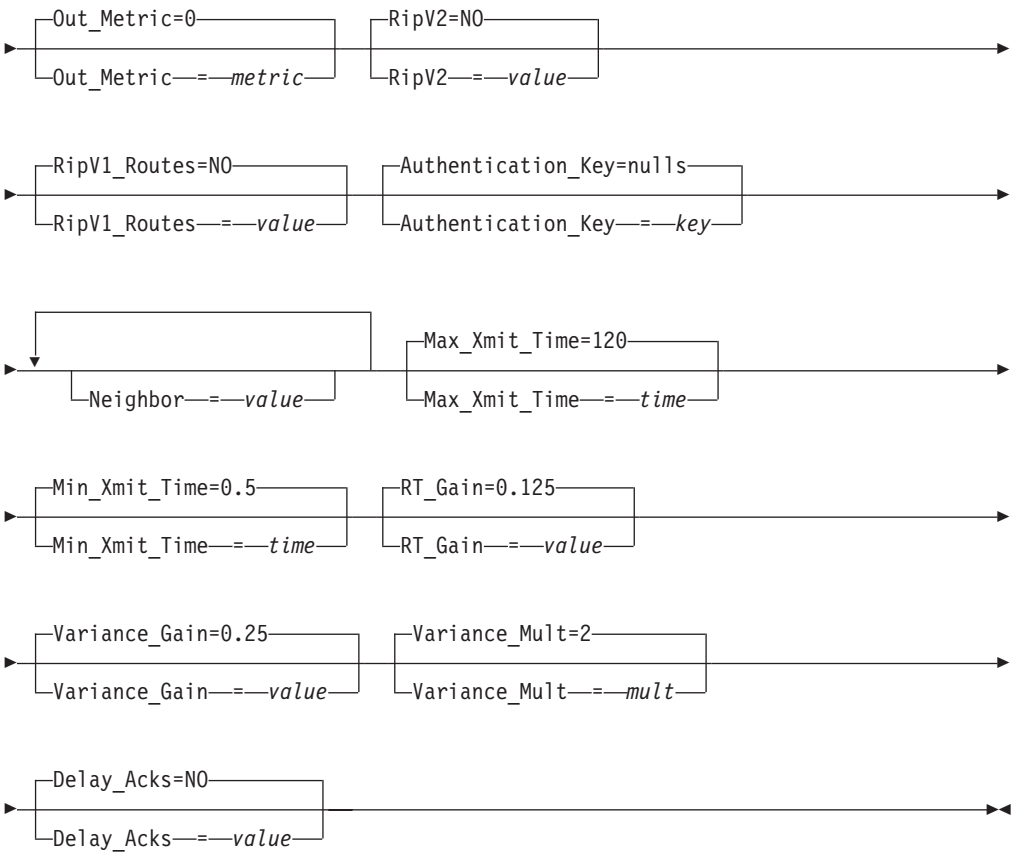
Specifies the cost that RIP will advertise with the default route that it originates. Valid values are 1 to 16.

RIP_INTERFACE

Configures the RIP parameters for each IP interface. This statement will need to be replicated in the config file for each IP interface over which RIP will operate.

Syntax:





Parameters:

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 wild cards. The valid wild-card specifications are below. The result of coding a wild card value are that all configured interfaces whose IP address matches the wild card will be configured as RIP interfaces. Configured interface IP address will be matched against possible wild cards in the order they appear below with x.y.z.* being the best match, x.y.*.* being 2nd best, and so forth.

```
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.*  - Same as ALL
ALL      - Same as *.*.*.*
```

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 profile. Valid values are any 16 characters. This parameter is ignored on wildcard interface definitions.

Subnet_Mask
Subnet mask for the associated interface IP address.

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 RIPV1 point-to-point links. If this parameter is not specified for a RIPV2 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 OMPROUTE initialization independent of whether this parameter is specified.

MTU
The maximum transmission unit size for RIP to add to the routing table for routes that take this interface. Valid values are 0 to 65,535.

Receive_RIP
Specifies whether or not to learn any RIP routes on this interface. Valid values are YES or NO. Note that when this parameter is coded as NO, only routes explicitly allowed via the Accept_RIP_Route configuration statement will be accepted on this interface.

Receive_Dynamic_Nets
Specifies whether or not to learn routes for networks over this interface. If this is not set, only nets explicitly allowed via 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 via 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 via the Accept_RIP_Route configuration statement will be accepted on this interface. Valid values are YES or NO.

filter
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 OMPROUTE 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 a 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 a 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 a 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 a 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 a RIP input filter.

receive_cond
Specifies that routes matching the dest_route and filter_mask are to be received over only this interface when that 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 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 (*) 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 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.

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.

Send_Only
Specifies broadcast restrictions. Valid values are VIRTUAL to broadcast only virtual IP addresses, DEFAULT to broadcast only the default route, DIRECT to broadcast only direct routes, TRIGGERED to broadcast only triggered updates, and ALL to specify no broadcast restrictions. More than one value can be specified (unless ALL is used) by separating the values with commas.

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 OMPROUTE 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, 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. In this context an appropriate subnet is one that meets RFC 1058 subnet advertisement constraints: - Natural Net must be the same as the IP source's natural net - Subnet mask must be the same - Valid values are YES or NO.

Send_Static_Routes
Advertise static and direct routes in RIP responses sent from this IP source address. 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.

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 on this link. Valid values are YES or NO.

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. Valid values are any alphanumeric string from code page 1047 up to 16 characters in length coded within double quotes or any hex string which begins with 0x.

Note: If the value is entered in characters (rather than the hex string), that value is case sensitive.

Neighbor
Multiple neighbor statements 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.

Max_Xmit_Time
The maximum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 999.990.

Min_Xmit_Time
The minimum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

RT_Gain
The round trip gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

Variance_Gain
The variance gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

Variance_Mult
The variance multiplier value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

Delay_Acks
The delay acknowledgments value to add to the routing table for routes that take this interface. Specifying YES delays transmission of acknowledgements when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgements being returned immediately. Valid values are YES and NO.

ACCEPT_RIP_ROUTE

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.

Syntax:

Parameters:

IP_address
Destination route to be unconditionally accepted.

FILTER

The filter statement can be coded stand-alone in the OMPROUTE configuration file (nosend and noreceive only) to apply to all configured RIP interfaces.

Syntax:

Parameters:

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 a 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.

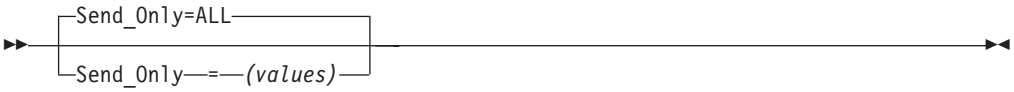
dest_route
The *dest_route* specifies the destination route in network, subnetwork, or host format in dotted decimal form. Alternatively, an asterisk (*) can be coded in conjunction with the nosend and noreceive filter types. This serves as a *blackhole* filter that can be used to filter out all routes broadcast 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.

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.

Send_Only

The Send_Only statement can be coded stand-alone in the OMPROUTE configuration file to apply to all RIP interfaces.

Syntax:



Parameters:

(values)
Specifies broadcast restrictions. Valid values are VIRTUAL to broadcast only virtual IP addresses, DEFAULT to broadcast only the default route, DIRECT to broadcast only direct routes, TRIGGERED to broadcast only triggered updates, and ALL to specify no broadcast restrictions. More than one value can be specified (unless ALL is used) by separating the values with commas.

Common Configuration Statements

This section contains descriptions of the common configuration statements:

- DEFAULT_ROUTE
- INTERFACE
- ROUTESA_CONFIG

DEFAULT_ROUTE

Allows the default route to be specified to OMPROUTE. Default routes are created using any of the following:

- GATEWAY statement
- Default_Route statement
- Originate_RIP_Default statement
- Learned by routing protocol

The Send_Default_Routes keyword on the RIP_Interface statement indicates whether or not to advertise the default route.

Syntax:

Parameters:

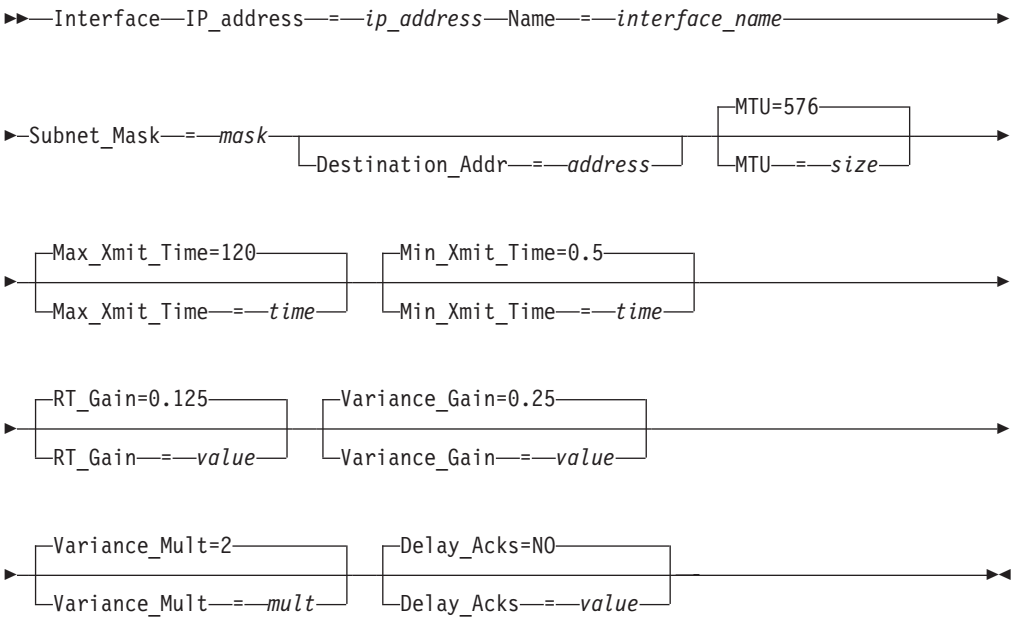
Name
The name of the interface. This name must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP profile. Valid values are any 16 characters.

Next_Hop
IP address of the next hop.

INTERFACE

Allows certain values to be specified for interfaces that are neither OSPF nor RIP interfaces. Each interface that is neither an OSPF nor a RIP interface should be configured to OMPROUTE via 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.

Syntax:



Parameters:

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 wild cards. The valid wild-card specifications are below. The result of coding a wild card value is that all configured interfaces whose IP address matches the wild card will be configured as interfaces. Configured interface IP address will be matched against possible wild cards in the order they appear below with x.y.z.* being the best match, x.y.*.* being 2nd best, and so forth.

```
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL     - Same as *.*.*.*
```

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 profile. Valid values are any 16 characters.

Subnet_Mask
Subnet mask for the associated interface's IP address.

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 OMPROUTE initialization independent of whether this parameter is specified.

MTU
The maximum transmission unit size for OMPROUTE to add to the routing table for routes that take this interface. Valid values are 0 to 65535.

Max_Xmit_Time
The maximum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 999.990.

Min_Xmit_Time
The minimum retransmit time to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

RT_Gain
The round trip gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

Variance_Gain
The variance gain value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 1.0.

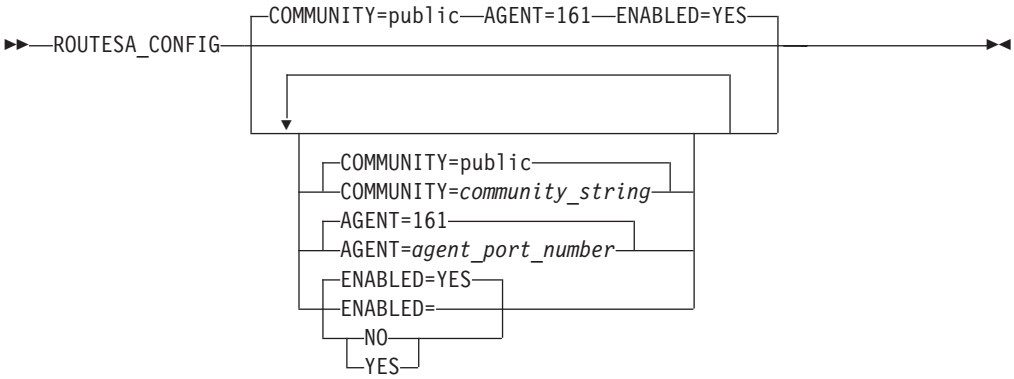
Variance_Mult
The variance multiplier value to add to the routing table for routes that take this interface. This value is used in the TCP/IP re-transmission time-out calculation to determine how long to wait for an acknowledgement before resending a packet. Valid values are 0 to 99.990.

Delay_Acks
The delay acknowledgments value to add to the routing table for routes that take this interface. Specifying YES delays transmission of acknowledgements when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgements being returned immediately. Valid values are YES and NO.

ROUTESA_CONFIG Statement

Use the ROUTESA_CONFIG statement to configure the OMPROUTE subagent.

Syntax:



Parameters:

COMMUNITY
A character string of 1–32 characters used as the community name (or password) in establishing contact with the SNMP agent. For the OMPROUTE subagent to communicate with the CS for OS/390 SNMP agent, the community name specified (or defaulted) on the COMMUNITY keyword must match one that is defined (or defaulted) in the PW.SRC or SNMPD.CONF data set configured to the SNMP agent or the -c parameter when the SNMP agent is started. The default value is *public*.

AGENT
A port number in the range 1–65535 used in establishing communication with the SNMP agent. For the OMPROUTE subagent to communicate with the CS for OS/390 SNMP agent, the port number specified must match the port number specified (or defaulted) on the -p parameter when the SNMP agent is started. The default value is 161.

ENABLED
A value of YES indicates that the OMPROUTE subagent should be started during OMPROUTE initialization. If there are no active OSPF interfaces, the OMPROUTE subagent will return *noSuchInstance* for all GET and GETNEXT requests. By default, the OMPROUTE subagent is started when OMPROUTE is started.

A value of NO indicates that the OMPROUTE subagent should not be started. Specify this keyword if little or no OSPF SNMP data will be requested from this OSPF image. SNMP MIB objects supported by the TCP/IP SNMP agent and TCP/IP subagent (other than the OMPROUTE subagent) will still be available. For information on which MIB objects are supported by the SNMP agent and OMPROUTE subagent, see the *OS/390 SecureWay Communications Server: IP User's Guide*.

Examples:

```
ROUTESA_CONFIG COMMUNITY USACCESS AGENT 528
ROUTESA_CONFIG ENABLED=NO
```

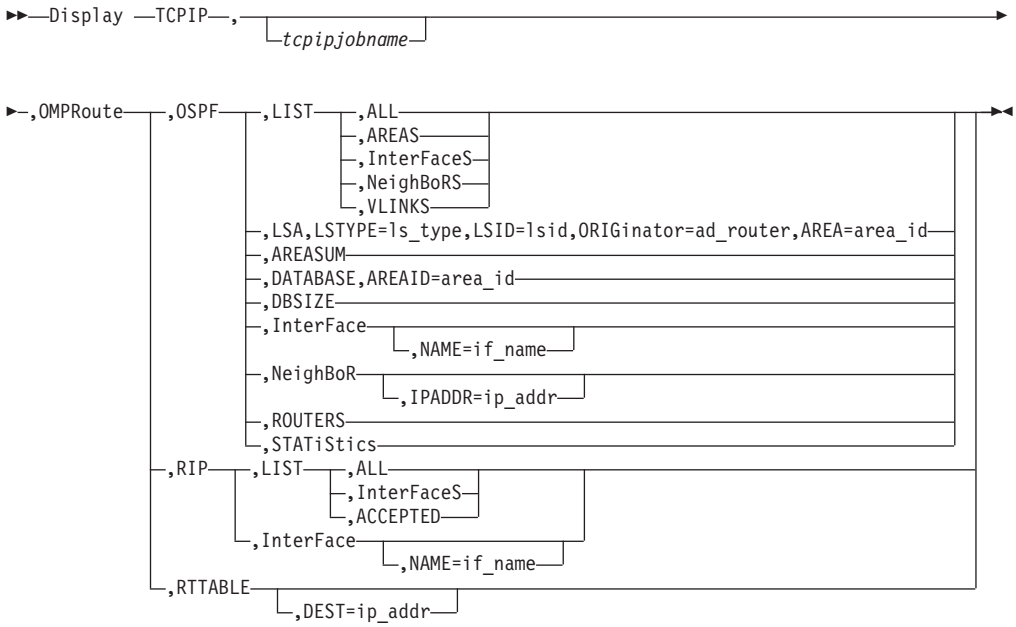
Usage Notes:

- If ENABLED=NO is specified, the OMPROUTE subagent will *NOT* be started during OMPROUTE initialization. If the ROUTESA_CONFIG statement itself is *NOT* specified, the OMPROUTE subagent will be started (this is the default).
- The community string is case sensitive and must be 1–32 characters. It is not converted to uppercase by profile processing.
- At initialization time, the default for the OMPROUTE subagent is to be enabled.
- A MODIFY command can be used to start or stop the OMPROUTE subagent, but the setting of the parameters cannot be changed unless OMPROUTE is recycled.

Using OMPROUTE DISPLAY Commands

You can use the DISPLAY command to display OSPF and RIP configuration and state information.

Syntax



The following sections provide details on the types of data that can be displayed as well as examples of the generated output.

All OSPF Configuration Information

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,ALL` command lists all OSPF-related information. A sample output with an explanation of entries follows:

```
EZZ7831I GLOBAL CONFIGURATION 967
    TRACE LEVEL: 1, DEBUG LEVEL: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:          TCPCS6
    OSPF PROTOCOL:           ENABLED
    EXTERNAL COMPARISON:     TYPE 1
    AS BOUNDARY CAPABILITY:  ENABLED
    IMPORT EXTERNAL ROUTES:  RIP SUB
    ORIG. DEFAULT ROUTE:     ALWAYS
    DEFAULT ROUTE COST:      (1, TYPE 2)
    DEFAULT FORWARD. ADDR:   9.167.100.17
    DEMAND CIRCUITS:         ENABLED

EZZ7832I AREA CONFIGURATION
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
2.2.2.2          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK            ADVERTISE?
2.2.2.2          9.167.200.0      255.255.255.0   YES
2.2.2.2          9.167.100.0      255.255.255.0   YES
```

```
EZZ78331 INTERFACE CONFIGURATION
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD
9.168.100.3     0.0.0.0             1     10      1     1     20     80
9.167.100.13    2.2.2.2             1     10      1     1     20     80

EZZ7836I VIRTUAL LINK CONFIGURATION
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD
9.67.100.8           2.2.2.2              20      5     40   160

EZZ7835I NBMA CONFIGURATION
                INTERFACE ADDR      POLL INTERVAL
                9.168.100.3         120

EZZ7834I NEIGHBOR CONFIGURATION
                NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
                9.168.100.56      9.168.100.3         YES
                9.168.100.70      9.168.100.3         NO
```

TRACE LEVEL
Displays the level of tracing currently in use by OMPROUTE.

DEBUG LEVEL
Displays the level of debugging currently in use by OMPROUTE.

SADEBUG LEVEL
Displays the level of debugging currently in use by OMPROUTE's OSPF SNMP subagent.

STACK AFFINITY
Displays the name of the stack on which OMPROUTE 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.

IMPORT EXTERNAL ROUTES
Indicates the types of external routes that will be imported into the OSPF domain. Displayed only when AS Boundary Capability is enabled.

ORIG DEFAULT ROUTE
Indicates whether the router will originate a default route into the OSPF domain. The Originate Default Route is displayed only when AS Boundary Capability is enabled.

DEFAULT ROUTE COST
Displays the cost and type of the default route (if advertised). The Default Route Cost is displayed only when AS Boundary Capability is enabled.

DEFAULT FORWARD ADDR
Displays the forwarding address specified in the default route (if advertised). The Default Forwarding Address is displayed only when AS Boundary Capability is enabled.

DEMAND CIRCUITS
Indicates whether demand circuit support is available for OSPF interfaces.

The remainder of the DISPLAY <TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,ALL output is described in the following sections:

Configured OSPF Areas and Ranges

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,AREAS command lists all information concerning configured OSPF areas and their associated ranges. A sample output with an explanation of entries follows:

```
EZZ7832I AREA CONFIGURATION 115
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
2.2.2.2          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK             ADVERTISE?
2.2.2.2          9.167.200.0      255.255.255.0    YES
2.2.2.2          9.167.100.0      255.255.255.0    YES
```

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.

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.

ADDRESS
Displays the network address for a given range within an area.

MASK Displays the subnet mask for a given range within an area.

ADVERTISE?
Indicates whether a given range within an area is to be advertised.

Configured OSPF Interfaces

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,INTERFACES command lists, for each OSPF interface, the IP address and configured parameters as coded in the OMPROUTE configuration file. (The keyword IFS can be substituted for INTERFACES.) A sample output with an explanation of entries follows:

```
EZZ7833I INTERFACE CONFIGURATION 165
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD
9.168.100.3     0.0.0.0             1     10      1     1     20     80
9.167.100.13    2.2.2.2             1     10      1     1     20     80
```

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.

TRNSDLY
Indicates the transmission delay, which is an estimate of the number of seconds required to transmit routing information over the interface. The number of seconds must be greater than zero.

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 Hellos cease to be heard that the router is declared down.

Configured OSPF Non-Broadcast, Multi-Access Networks

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,NBMA` command lists the interface address and polling interval related to interfaces connected to non-broadcast, multi-access networks. A sample output follows:

```
EZZ7835I NBMA CONFIGURATION 191
               INTERFACE ADDR      POLL INTERVAL
               9.168.100.3         120
```

Configured OSPF Virtual Links

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,VLINKS` command lists all virtual links that have been configured with this router as the endpoint. A sample output with an explanation of entries follows:

```
EZZ7836I VIRTUAL LINK CONFIGURATION 202
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD
9.67.100.8           2.2.2.2              20      5     40   160
```

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. The number of seconds must be greater than zero.

HELLO
Indicates the number of seconds between Hello Packets sent from the interface.

DEAD Indicates the number of seconds after Hellos cease to be heard that the router is declared down.

Configured OSPF Neighbors

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,LIST,NEIGHBORS command lists the neighbors to non-broadcast networks. (The keyword NBRS can be substituted for NEIGHBORS.) A sample output with an explanation of entries follows:

```
EZZ7834I NEIGHBOR CONFIGURATION 205
                NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
                9.168.100.56      9.168.100.3         YES
                9.168.100.70      9.168.100.3         NO
```

NEIGHBOR ADDR
Indicates the IP address of the neighbor.

INTERFACE ADDRESS
Indicates the IP address of the interface to the neighbor.

DR ELIGIBLE?
Indicates whether the neighbor is eligible to become the designated router on the net.

OSPF Link State Advertisement

The following command displays the contents of a single link state advertisement contained in the OSPF database:

```
DISPLAY TCPIP,,OMPROUTE,OSPF,LSA,LSTYPE=ls-type,LSID=lsid,ORIG=ad-router,AREAID=area-id
```

For a summary of all the advertisements in the OSPF database, use the following command:

```
DISPLAY <TCPIP,<tcpipjobname>,OMPROUTE,OSPF,DATABASE,AREAID=<area-id>
```

A link state advertisement is defined by its link state type, link state ID and its advertising router. There is a separate link state database for each OSPF area. Providing an 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 (link-state-type = 1)
Describe the collected states of a router's interfaces

Network links (link-state-type = 2)
Describe the set of routers attached to a network

Summary link, IP network (link-state-type = 3)
Describe inter-area routes to networks

Summary link, ASBR (link-state-type = 4)
Describe inter-area routes to AS boundary routers

AS external link (link-state-type = 5)
Describe 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

```
EZZ7880I LSA DETAILS 220
        LS AGE:          292
        LS OPTIONS:      E,DC
        LS TYPE:         1
        LS DESTINATION (ID): 9.167.100.13
        LS ORIGINATOR:   9.167.100.13
        LS SEQUENCE NO:  0X80000009
        LS CHECKSUM:     0X8F78
        LS LENGTH:       36
        ROUTER TYPE:  ABR
        # ROUTER IFCS:   1
              LINK ID:          9.67.100.8
              LINK DATA:        9.167.100.13
              INTERFACE TYPE:   4
                      NO. OF METRICS: 0
                      TOS 0 METRIC:   2 (2)
```

LS AGE
Indicates the age of the advertisement in seconds.

LS OPTIONS
Indicates the optional OSPF capabilities supported by the piece of the routing domain described by the advertisement. 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 (can forward IP multicast datagrams), and DC (can support demand circuits).

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/subnet number.

LS ORIGINATOR
OSPF router ID of the originating router.

LS SEQUENCE NUMBER
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 one 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. ASBR means that the router is an AS boundary router, ABR that the router is an area border router, and W that the router is a wildcard multicast receiver.

 # 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's 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's network/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), or 4 (virtual link).

NO. OF METRICS
The number of nonzero TOS values for which metrics are provided for this interface. For the OS/390 implementation, this value will always be zero.

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 `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,AREASUM` command displays the statistics and parameters for all OSPF areas attached to the router. A sample output with an explanation of entries follows:

```
EZZ7848I AREA SUMMARY 222
AREA ID        AUTHENTICATION   #IFCS  #NETS  #RTRS  #BRDRS DEMAND
0.0.0.0           NONE              2      0      2      2  ON
2.2.2.2           NONE              1      0      3      2  ON
```

AREA ID
Indicates the ID of the area.

AUTHENTICATION
Indicates the authentication method being used by 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 DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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 five for AS external advertisements)
- Its link state ID (called the LS destination)
- The advertising router (called the LS originator)

A sample output with an explanation of entries follows:

```
EZZ7853I AREA LINK STATE DATABASE 269
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  5 @9.67.100.0         9.67.100.8      0X80000001    4  0X408
  5 @9.169.100.0        9.67.100.8      0X80000001    4  0X73E
  5 @9.169.100.14       9.67.100.8      0X80000001    4  0XE66
  5 @192.8.8.0          9.67.100.8      0X80000001    4  0XAAF
  5 @192.8.8.8          9.67.100.8      0X80000001    4  0X5A4
              # ADVERTISEMENTS:       5
              CHECKSUM TOTAL:         0X2A026
```

TYPE Always 5 for AS external advertisements.

LS DESTINATION
Indicates an IP network/subnet number. These network numbers belong to other Autonomous Systems.

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.

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 OSPF routers have synchronized databases.

OSPF Area Link State Database

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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)

A sample output with an explanation of entries follows:

```
EZZ7853I AREA LINK STATE DATABASE 352
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  1 @9.67.100.7         9.67.100.7      0X80000016  113  0X5D8D
  1 @9.67.100.8         9.67.100.8      0X80000014   88  0XC0AE
  1 @9.167.100.13       9.167.100.13    0X80000013  100  0X4483
  3 @9.167.100.13       9.167.100.13    0X80000001  760  0XF103
               # ADVERTISEMENTS:       4
               CHECKSUM TOTAL:         0X253C1
```

TYPE Separate LS types are numerically displayed: type 1 (router links advertisements), type 2 (network links advertisements), type 3 (network summaries), and type 4 (AS boundary router summaries).

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.

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 Interface Statistics and Parameters

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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. Sample outputs with explanations of entries follow:

```
----  Example 1  ----
EZZ7849I INTERFACES 354
IFC ADDRESS     PHYS      ASSOC. AREA    TYPE   STATE    #NBRS    #ADJS
9.168.100.3     CTC1      0.0.0.0         P-P      16        0        0
9.167.100.13    CTC2      2.2.2.2         P-P      16        1        1
UNNUMBERED      VL/0      0.0.0.0         VLINK    16        1        1
```

IFC ADDRESS
Interface IP address.

PHYS Displays the interface name.

ASSOC AREA
Attached area ID.

TYPE Can be Brdcst (broadcast, for example, an Ethernet interface), P-P (a point-to-point network—for example, a synchronous serial line), Multi (non-broadcast, multi-access—for example, an X.25 connection), or VLink (an OSPF virtual link).

STATE
Can be one of the following: 1 (down), 2 (backup), 4 (looped back), 8 (waiting), 16 (point-to-point), 32 (DR other), 64 (backup DR), or 128 (designated router). For more information about these values, refer to RFC 1583.

#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.

```
----  Example 2  ----
EZZ7850I INTERFACE DETAILS 356
               INTERFACE ADDRESS:      9.168.100.3
               ATTACHED AREA:          0.0.0.0
               PHYSICAL INTERFACE:     CTC1
               INTERFACE MASK:         255.255.255.0
               INTERFACE TYPE:         P-P
               STATE:                  16
               DESIGNATED ROUTER:      0.0.0.0
               BACKUP DR:              0.0.0.0

DR PRIORITY:       1  HELLO INTERVAL:   20  RXMT INTERVAL:    10
DEAD INTERVAL:    80  TX DELAY:          1  POLL INTERVAL:     0
DEMAND CIRCUIT:  OFF  HELLO SUPPRESS:  OFF  SUPPRESS REQ:    OFF
MAX PKT SIZE:    556  TOS 0 COST:        1

# NEIGHBORS:       0  # ADJACENCIES:     0  # FULL ADJS.:      0
# MCAST FLOODS:    0  # MCAST ACKS:      0
MC FORWARDING:   OFF  DL UNICAST:      OFF
NETWORK CAPABILITIES:
 POINT-TO-POINT
```

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 either Brdcst (broadcast—for example, an Ethernet interface), P-P (a point-to-point network—for example, a synchronous serial line), Multi (non-broadcast, multi-access—for example, an X.25 connection) or VLink (an OSPF virtual link).

STATE
Can be one of the following: 1 (down), 2 (backup), 4 (looped back), 8 (waiting), 16 (point-to-point), 32 (DR other), 64 (backup DR), or 128 (designated router). See STATE above for more details.

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.

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.

 # 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).

 # MCAST FLOODS
Number of link state updates flooded out the interface (not counting retransmissions).

 # MCAST ACKS
Number of link state acknowledgments flooded out the interface (not counting retransmissions).

NETWORK CAPABILITIES
Displays the capabilities of the interface.

OSPF Neighbor Statistics and Parameters

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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. Following are sample outputs with explanations of entries:

```
----  Example 1  ----
EZZ7851I NEIGHBOR SUMMARY 358
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
9.167.100.17    9.67.100.7        128     0     0     0  OFF CTC2
VL/0            9.67.100.8        128     0     0     0  OFF *
```

NEIGHBOR ADDR
Displays the neighbor 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), or 128 (Full). For more information about these values, refer to RFC 1583.

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 more recent advertisements that are being requested from the neighbor.

HSUP Displays whether Hello Suppression is active with the neighbor.

IFC Displays the interface shared by the router and the neighbor.

```
----  Example 2 ----
EZZ7852I NEIGHBOR DETAILS 360
                NEIGHBOR IP ADDRESS:    9.167.100.17
                OSPF ROUTER ID:         9.67.100.7
                NEIGHBOR STATE:         128
                PHYSICAL INTERFACE:     CTC2
                DR CHOICE:              0.0.0.0
                BACKUP CHOICE:          0.0.0.0
                DR PRIORITY:            1
                NBR OPTIONS:            E
 DB SUMM QLEN:      0  LS RXMT QLEN:      0  LS REQ QLEN:       0
 LAST HELLO:        1  NO HELLO:        OFF
 # LS RXMITS:       1  # DIRECT ACKS:     2  # DUP LS RCVD:     2
 # OLD LS RCVD:     0  # DUP ACKS RCVD:   0  # NBR LOSSES:      0
 # ADJ. RESETS:     2
```

NEIGHBOR IP ADDRESS
Neighbor 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), or 128 (Full).

PHYSICAL INTERFACE
Displays interface name of the router and neighbor common interface.

DR CHOICE, BACKUP CHOICE, DR PRIORITY
Indicate the values seen in the last hello received from the neighbor.

NBR OPTIONS
Indicates the optional OSPF capabilities supported by 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), and 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 zero 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 entries to state ExStart.

OSPF Router Routes

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,ROUTERS` command displays all routes to other routers that have been calculated by OSPF and are now present in the routing table. A sample output with explanations of entries follows:

```
EZZ7855I OSPF ROUTERS 362
DTYPE RTYPE DESTINATION       AREA           COST       NEXT HOP(S)
  BR  SPF   9.67.100.8       2.2.2.2          2        9.167.100.17
  BR  SPF   9.67.100.8       0.0.0.0          2        9.67.100.8
ASBR  SPF   9.67.100.8       2.2.2.2          2        9.167.100.17
```

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's OSPF identification.

AREA Displays the AS area to which it 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 Link State Database Statistics

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,OSPF,DBSIZE command displays the number of LSAs currently in the link state database, categorized by type. The following is a sample output:

```
EZZ7854I LINK STATE DATABASE SIZE 364
               # ROUTER-LSAS:            5
               # NETWORK-LSAS:           0
               # SUMMARY-LSAS:           7
               # SUMMARY ROUTER-LSAS:    1
               # AS EXTERNAL-LSAS:       5
               # INTRA-AREA ROUTES:      4
               # INTER-AREA ROUTES:      0
               # TYPE 1 EXTERNAL ROUTES: 5
               # TYPE 2 EXTERNAL ROUTES: 0
```

OSPF Routing Protocol Statistics

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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. A sample output with explanations of entries follow:

```
EZZ7856I OSPF STATISTICS 366
               OSPF ROUTER ID:         9.167.100.13
               EXTERNAL COMPARISON:    TYPE 1
               AS BOUNDARY CAPABILITY: NO
               IMPORT EXTERNAL ROUTES: NONE
               ORIG. DEFAULT ROUTE:    NO
               DEFAULT ROUTE COST:     (1, TYPE 2)
               DEFAULT FORWARD. ADDR.: 0.0.0.0

ATTACHED AREAS:                 2  OSPF PACKETS RCVD:              194
OSPF PACKETS RCVD W/ERRS:       1  TRANSIT NODES ALLOCATED:         82
TRANSIT NODES FREED:           77  LS ADV. ALLOCATED:               53
```

```
LS ADV. FREED:                  40  QUEUE HEADERS ALLOC:             32
QUEUE HEADERS AVAIL:            32  MAXIMUM LSA SIZE:               528
# DIJKSTRA RUNS:                25  INCREMENTAL SUMM. UPDATES:       0
INCREMENTAL VL UPDATES:          0
MULTICAST PKTS SENT:           227  UNICAST PKTS SENT:               36
LS ADV. AGED OUT:                0  LS ADV. FLUSHED:                 10
PTRS TO INVALID LS ADV:          0  INCREMENTAL EXT. UPDATES:        19
```

OSPF ROUTER ID
Displays the router OSPF ID.

EXTERNAL COMPARISON
Displays the external route type used by the router when importing external routes.

AS BOUNDARY CAPABILITY
Displays whether external routes will be imported.

IMPORT EXTERNAL ROUTES
Displays the external routes that will be imported.

ORIG. DEFAULT ROUTE
Displays whether the router will advertise an OSPF default route.

DEFAULT ROUTE COST
Displays the cost and type of the default route (if advertised).

DEFAULT FORWARD ADDR
Displays the forwarding address specified in the default route (if advertised).

ATTACHED AREAS
Indicates the number of areas that the router has active interfaces to.

OSPF PACKETS RCVD
Covers all types of OSPF protocol packets.

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 freed, 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
Indicate 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.

INCREMENTAL EXT. UPDATES
Displays the number of changes to external destinations that are incrementally installed in the routing table.

OMPROUTE Routing Table

The `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,RTTABLE` command displays all of the routes in the OMPROUTE routing table. A sample output with explanation of entries follow.

Note: Be aware that this command displays the contents of the working table that is used by OMPROUTE, not the TCP/IP routing table. The contents of the OMPROUTE routing table may contain information different from that in the TCP/IP routing table.

```
EZZ7847I ROUTING TABLE 368
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SBNT   9.0.0.0          FF000000  1       576     NONE
SPE1   9.67.100.0       FFFFFF00  3       571     9.167.100.17
 SPF   9.67.100.7       FFFFFFFF  3       595     9.167.100.17
 SPF   9.67.100.8       FFFFFFFF  2       1270    9.167.100.17
 DIR*  9.167.0.0        FFFF0000  1       1685    9.167.100.13
 SPF   9.167.100.13     FFFFFFFF  2       1292    CTC2
 SPF*  9.167.100.17     FFFFFFFF  1       1684    9.167.100.17
 DIR*  9.168.100.0      FFFFFF00  1       1686    9.168.100.3
 DIR*  9.168.100.4      FFFFFFFF  1       1686    9.168.100.3
SPE1   9.169.100.0      FFFFFF00  3       571     9.167.100.17
SPE1   9.169.100.14     FFFFFFFF  3       571     9.167.100.17
SPE1   192.8.8.0        FFFFFF00  3       571     9.167.100.17
SPE1   192.8.8.8        FFFFFFFF  3       572     9.167.100.17
                                 0 NETS DELETED, 3 NETS INACTIVE
```

TYPE Indicates how the route was derived:

SBNT Indicates that the network is subnetted; such an entry is a placeholder only.

DIR Indicates a directly connected network or subnet.

RIP Indicates a route that was learned through the RIP protocol.

DEL Indicates the route has been deleted.

STAT Indicates a statically configured route.

SPF Indicates that the route is an OSPF intra-area route.

SPIA Indicates that the route is an OSPF inter-area route.

SPE1, SPE2
Indicates OSPF external routes (type 1 and 2, respectively).

RNGE Indicates a route type that is an active OSPF area address range and is not used in forwarding packets.

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 network/subnet.

DEST NET
Indicates the IP destination.

MASK Indicates the IP destination subnet mask.

COST Indicates the route cost.

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 host. A number in parentheses at the end of the column indicates the number of equal-cost routes to the destination. Use the `DISPLAY <TCPIP,<tcpipjobname>,OMPROUTE,RTTABLE,dest=<ip-addr>` command to obtain a list of the next hops.

Route Expansion Information

Use the `DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,RTTABLE,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. A sample output with explanation of entries follows:

```
EZZ7874I ROUTE EXPANSION 370
DESTINATION:    9.68.101.0
MASK:           255.255.255.0
ROUTE TYPE:     SPF
DISTANCE:       6
AGE:            1344
NEXT HOP(S):    9.167.100.17      (CTC2)
                9.168.100.4       (CTC1)
```

DESTINATION
Indicates the IP destination.

MASK Indicates the IP destination subnet mask.

ROUTE TYPE
Indicates how the route was derived:

SBNT Indicates that the network is subnetted; such an entry is a placeholder only.

DIR Indicates a directly connected network or subnet.

RIP Indicates a route that was learned through the RIP protocol.

DEL Indicates the route has been deleted.

STAT Indicates a statically configured route.

SPF Indicates that the route is an OSPF intra-area route.

SPIA Indicates that the route is an OSPF inter-area route.

SPE1, SPE2
Indicates OSPF external routes (type 1 and 2, respectively).

RNGE Indicates a route type that is an active OSPF area address range and is not used in forwarding packets.

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 network/subnet.

DISTANCE
Indicates the route cost.

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 host.

RIP Configuration Information

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,RIP,LIST,ALL command lists all RIP-related configuration information. A sample output with explanations of entries follows:

```
EZZ7843I RIP CONFIGURATION 447
TRACE LEVEL: 1, DEBUG LEVEL: 0, SADEBUG LEVEL: 0
STACK AFFINITY:  TCPCS6
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC2             9.167.100.13    RIP VERSION 1
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0
CTC1             9.168.100.3     RIP VERSION 1
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0

EZZ7844i RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  9.167.100.79         9.167.100.59
```

TRACE LEVEL
Displays the level of tracing currently in use by OMPROUTE.

DEBUG LEVEL
Displays the level of debugging currently in use by OMPROUTE.

SADEBUG LEVEL
Displays the level of debugging currently in use by OMPROUTE's OSPF SNMP subagent.

STACK AFFINITY
Displays the name of the stack on which OMPROUTE is running.

The remainder of the DISPLAY <TCPIP,<tcpipjobname>,OMPROUTE,RIP,LIST,ALL output is described in the following sections.

Configured RIP Interfaces

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,RIP,LIST,INTERFACES command lists IP addresses and configured parameters for each RIP interface. (The keyword IFS can be substituted for INTERFACES.) A sample output with explanations of entries follows:

```
EZZ7843I RIP CONFIGURATION 447
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC2             9.167.100.13    RIP VERSION 1
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0
CTC1             9.168.100.3     RIP VERSION 1
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0
```

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 communicated over this interface.

SEND Specifies which types of routes will be included in RIP responses sent out 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 Routes to Be Accepted

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,RIP,LIST,ACCEPTED command lists the routes to be unconditionally accepted, as configured with the ACCEPT_RIP_ROUTE statement. A sample output follows:

```
EZZ7844I RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  9.167.100.79       9.167.100.59
```

ACCEPT RIP UPDATES ALWAYS FOR
Indicates for which destination network, networks, subnet, or subnets RIP updates are always accepted.

RIP Interface Statistics and Parameters

The DISPLAY TCPIP,<tcpipjobname>,OMPROUTE,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 (DISPLAY <TCPIP,<tcpipjobname>,OMPROUTE,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.)

```
----  Example 1  ----
EZZ78591 RIP INTERFACES 464
IFC ADDRESS     IFC NAME      SUBNET MASK     MTU    DESTINATION
9.167.100.13    CTC2          255.255.0.0     576    9.167.100.17
```

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.

```
---- Example 2  ----
EZZ7860I RIP INTERFACE DETAILS 466
INTERFACE ADDRESS:      9.167.100.13
INTERFACE NAME:         CTC2
SUBNET MASK:            255.255.0.0
MTU                     576
DESTINATION ADDRESS:    9.167.100.17
RIP VERSION:            1
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 POIS. REV. ROUTES: YES

SEND ONLY: VIRTUAL, DEFAULT

FILTERS: SEND      9.67.100.0    255.255.255.0
         RECEIVE   9.67.101.0    255.255.255.0
```

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-ot-point.

RIP VERSION
Indicates whether RIP Version 1 or RIP Version 2 packets are communicated over this route.

IN METRIC
Indicates the RIP interface input metric.

OUT METRIC
Indicates the RIP interface output metric

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 route.

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 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).

SEND ONLY
Indicates the route-type restrictions on RIP broadcasts for this interface.

FILTERS
Indicates the send and receive filters for this interface.

Global RIP Filters

The `DISPLAY TCPIP,<tcpipjobname>,RIP,FILTERS` command displays the Global RIP filters. A sample output with explanations of entries follows.

```
EZZ78012I GLOBAL RIP FILTERS

SEND ONLY: VIRTUAL, DEFAULT
```

```
FILTERS: SEND       9.67.100.0    255.255.255.0
         NORECEIVE  9.67.101.0    255.255.255.0
```

SEND ONLY
Indicates the global route-type restrictions on RIP broadcasts that apply to all RIP interfaces.

FILTERS
Indicates the global send and receive filters that apply to all RIP interfaces.

Configuring Interfaces to OMPROUTE

You can use three different configuration statements to configure interfaces to OMPROUTE. These statements are:

- OSPF_INTERFACE
- RIP_INTERFACE
- INTERFACE

Follow these guidelines to decide which configuration statement to use:

- Use OSPF_INTERFACE if the OSPF protocol will be communicated with one or more routers over the interface.
- Use RIP_INTERFACE if the RIP protocol will be communicated with one or more routers over the interface.
- Use INTERFACE if neither the OSPF nor the RIP protocol will be communicated with other router or routers over the interface. An exception to this guideline occurs when the interface being configured is a virtual (VIPA) interface. For more information on this exception, see “Configuring VIPA Interfaces” on page 1017.

After selecting which configuration statement to use, you need to correctly define the selected statement. The interface type determines the correct method of configuration. Configuring the various types of interfaces is discussed in the following sections.

Each of these sections explain which parameters are required for each of the three interface configuration statements. Also included is a sample for each of the three configuration statements.

Notes:

1. Although many parameters exist, not all are included on the sample configuration statements presented in this section. The excluded parameters are being allowed to take their default values. In most cases, this is acceptable. In other cases, you might want to include some of the additional parameters for a particular interface to refine its configuration.
2. All of the sample OSPF_INTERFACE statements include the parameters ATTACHES_TO_AREA and COST0. Although these are not required parameters, they are included in the samples because they frequently need to be specified. All of the other parameters used in the sample OSPF_INTERFACE statements are required for that type of interface.
3. All of the sample RIP_INTERFACE statements include the parameter RIPV2=YES. These samples assume that you are using RIP Version 2. If you are using RIP Version 1, remove the RIPV2 parameter.

Configuring Point-to-Point Interfaces

Point-to-point interfaces are interfaces such as Channel-to-Channel (CTC) and Claw interfaces. For these interfaces, specify the DESTINATION_ADDR parameter to allow for the creation of a host route to the address at the remote end of the interface. The following examples show various interfaces configurations.

Sample OSPF_INTERFACE

```
OSPF_Interface
  IP_Address=9.67.113.80
  Name=CTCD00
  Subnet_Mask=255.255.255.254
  Destination_Addr=9.67.113.81
  Attaches_To_Area=1.1.1.1
  Cost0=5;
```

Sample RIP_INTERFACE

```
RIP_Interface
  IP_Address=9.67.114.80
  Name=CTCA00
  Subnet_Mask=255.255.255.254
  Destination_Addr=9.67.114.81
  RIPV2=Yes;
```

Sample INTERFACE

```
Interface
   IP_Address=9.17.214.10
   Name=CTCB00
   Subnet_Mask=255.255.255.254
   Destination_Addr=9.17.214.11;
```

Note: If another router is directly attached via a Claw device, and the OSPF protocol is being communicated with that router, the other router must also be configured to view the Claw device as a point-to-point interface. Failure to do this results in a failure to add any routes via that router.

Configuring MPC, XCF and IUT SameHost Interfaces

If the OSPF or RIP protocol will be communicated with one or more routers over an MPX, XCF, or IUT interface, OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate the OSPF or RIP packets. However, due to underlying signalling that takes place when a host connects to these network types, the TCP/IP stack is able to ″learn″ the required IP addresses. In turn, OMPROUTE learns those IP addresses from the TCP/IP stack. As a result, it is *not* necessary to configure the neighbor addresses on the OSPF_INTERFACE and RIP_INTERFACE statements.

Sample OSPF_INTERFACE

```
OSPF_Interface
   IP_Address=9.27.13.81
   Name=XCFD00
   Subnet_Mask=255.255.255.0
   Attaches_To_Area=1.1.1.1
   Cost0=7;
```

Sample RIP_INTERFACE

```
RIP_Interface
   IP_Address=9.46.33.181
   Name=MPCA01
   Subnet_Mask=255.255.255.0
   RIPV2=Yes;
```

Sample INTERFACE

```
Interface
    IP_Address=9.77.13.49
    Name=XCFB00
    Subnet_Mask=255.255.255.0;
```

Configuring Non-Broadcast Network Interfaces

Non-broadcast network interfaces are interfaces such as Hyperchannel and ATM. If the OSPF or RIP protocol will be communicated with one or more routers over a non-broadcast network interface, OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate the OSPF or RIP packets. For non-broadcast network interface types, there is no underlying signalling that allows the TCP/IP stack to ″learn″ the required IP addresses (as there is with MPC, XCF, and IUT SameHost). As a result, the neighbor addresses must be configured to OMPROUTE (with the DR_NEIGHBOR and/or NO_DR_NEIGHBOR parameters on the OSPF_INTERFACE statement and with the NEIGHBOR parameter on the RIP_INTERFACE statement). In addition, the OSPF_INTERFACE statement must include the NON_BROADCAST=YES parameter to indicate that it is a non-broadcast, multi-access interface and the ROUTER_PRIORITY parameter to assist in electing a Designated Router for the network.

Sample OSPF_INTERFACE

```
OSPF_Interface
   IP_Address=9.37.84.49
   Name=HCHE00
   Subnet_Mask=255.255.255.0
   Attaches_To_Area=0.0.0.0
   Cost0=3
   Router_Priority=2
   Non_Broadcast=Yes
   DR_Neighbor=9.37.84.53
   No_DR_Neighbor=9.37.84.63;
```

Sample RIP_INTERFACE

```
RIP_Interface
   IP_Address=9.37.104.79
   Name=ATME00
   Subnet_Mask=255.255.255.0
   RIPV2=Yes
   Neighbor=9.37.104.53
   Neighbor=9.37.104.85;
```

Sample INTERFACE

```
Interface
   IP_Address=9.77.13.49
   Name=ATMB00
   Subnet_Mask=255.255.255.0;
```

Configuring Broadcast Network Interfaces

Broadcast Network interfaces are interfaces such as Token Ring, Ethernet, and FDDI. These network media allow for the broadcasting and multicasting of packets. Therefore, it is not necessary for OMPROUTE to know the IP addresses of the other routers on the network in order for OSPF or RIP packets to be communicated with those routers. OMPROUTE sends OSPF/RIP packets to the other routers on the network by using the appropriate broadcast (RIPv1) or multicast (OSPF and RIPv2) address. The IP addresses of the other routers are ″learned″ as OSPF/RIP packets are received from them. The OSPF_INTERFACE statement for a broadcast network 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=TR1
   Subnet_Mask=255.255.255.0
   Attaches_To_Area=0.0.0.0
   Cost0=2
   Router_Priority=1;
```

Sample RIP_INTERFACE

```
RIP_Interface
  IP_Address=9.29.107.3
  Name=TR2
  Subnet_Mask=255.255.255.0
  RIPV2=Yes;
```

Sample INTERFACE

```
Interface
   IP_Address=9.77.13.49
   Name=ETHB00
   Subnet_Mask=255.255.255.0;
```

Note: In the absence of the required multicast capability, it is possible to configure the OSPF_INTERFACE or RIP_INTERFACE statements as if the interface were a Non-Broadcast Network interface. This would cause OMPROUTE 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.

Configuring VIPA Interfaces

Special Considerations for Dynamic VIPA

The names of Dynamic VIPA interfaces are assigned programmatically by the stack when a Dynamic VIPA interface is created. Therefore, the Name field set on the Interface or OSPF_Interface statement for a Dynamic VIPA will be ignored by OMPROUTE. It is recommended that a host have an Interface or OSPF_Interface definition for every Dynamic VIPA address which that host might own. Because this could be a large number of interfaces, additional wildcard capabilities are added to OMPROUTE (for Dynamic VIPA interfaces only).

Ranges of Dynamic VIPA interfaces can be defined using the subnet mask parameter on the OSPF_Interface or Interface statement. The range defined in this way will be all the IP addresses that fall within the subnet defined by the mask and the IP address. For example:

```
OSPF_Interface
   IP_address = 10.138.165.80
   Name = whatever
   Subnet_mask = 255.255.255.240;
```

This example defines a range of Dynamic VIPA addresses from 10.138.65.80 to 10.138.165.95. For consistency with the VIPARANGE statement in the TCP/IP profile, any value that might fall within the range can be used with the mask to define a range of Dynamic VIPAs.

The following interface statement has the same meaning as the one in the example shown above:

```
OSPF_Interface
   IP_address = 10.138.165.87
   Name = whatever
   Subnet_mask = 255.255.255.240;
```

Notes:

1. When defining ranges, it is not necessary or desirable to code a destination address. OMPROUTE will automatically set the destination address of a Dynamic VIPA to its IP address.
2. There is nothing in the interface definition statements that informs OMPROUTE that a particular interface definition statement is for a Dynamic VIPA or a range of Dynamic VIPAs. Rather, OMPROUTE learns this information from the stack when these interfaces are created or taken over.

VIPA interfaces are the exception to the typical guidelines for choosing which type of configuration statement (OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE) to use. If the OSPF protocol *is not* being used (on any interfaces), then the typical guidelines apply. Because neither the OSPF protocol nor the RIP protocol is being communicated over the VIPA interface, the INTERFACE statement is used to configure the VIPA to OMPROUTE. Following is an example of this type of interface:

Sample INTERFACE

```
Interface
   IP_Address = 10.138.165.9
   Name = VLINK1
   Subnet_Mask = 255.255.255.248
   Destination_Addr = 10.138.165.9;
```

If the OSPF protocol *is* used (on any interfaces), then the typical guidelines do not apply. In this case, use the OSPF_INTERFACE statement to configure VIPA interfaces to OMPROUTE. Following is an example of this type of interface:

Sample OSPF_INTERFACE

```
OSPF_Interface
   IP_address = 10.138.165.9
   Name = VLINK1
   Subnet_mask = 255.255.255.248
   Destination_Addr =10.138.165.9
   Attaches_To_Area=1.1.1.1;
```

Notes:

1. The reason for not adhering to the typical guideline when the OSPF protocol is being used is that non-OSPF information can only be advertised by your router if your router is configured as an AS Boundary Router. Under certain circumstances (for example, when your router resides in an OSPF stub area), your router cannot be configured as an AS Boundary Router. Using the OSPF_Interface statement to configure your VIPA interfaces allows the VIPA information to be advertised regardless of stub area status.
2. For both of the above methods of configuring VIPA interfaces, set the DESTINATION_ADDR parameter to the same value as the IP_ADDRESS parameter. This is because there really is no other destination with a VIPA.

Chapter 27. Using OSPF

This chapter describes how to use the Open Shortest Path First (OSPF) Protocol.

OSPF Routing Summary

When a router is initialized, it uses the Hello Protocol to send Hello packets to its neighbors, and they in turn send their packets to the router. On broadcast and point-to-point networks, the router dynamically detects its neighboring routers by sending the Hello packets to the multicast address *ALLSPFRouters (224.0.0.5);* on non-broadcast networks you must configure information to help the router discover its *neighbors*. On all multi-access networks (broadcast and non-broadcast), the Hello Protocol also elects a *designated router* for the network.

Note: ATM native networks allow IP to use the network as a Non-Broadcast Multiple Access network. Thus, OSPF should be configured assuming non-broadcast. If you are using LAN Emulation, the network is treated as a broadcast network, and you should configure OSPF accordingly.

The router then attempts to form adjacencies with its neighbors to synchronize their topological databases. Adjacencies control the distribution (sending and receiving) of the routing protocol packets as well as the distribution of the topological database updates. On a multi-access network, the designated router determines which routers become adjacent.

A router periodically advertises its status or link state to its adjacencies. *Link state advertisements* (LSAs) flood throughout an area, ensuring that all routers have exactly the same topological database. This database is a collection of the link state advertisements received from each router belonging to an area. From the information in this database, each router can calculate a shortest path tree with itself designated as the root. Then the shortest path tree generates the routing table.

OSPF includes the following features:

- *Least-Cost Routing.* Allows you to configure path costs based on any combination of network parameters. For example, bandwidth, delay, and dollar cost.
- *No limitations to the routing metric.* While RIP restricts the routing metric to 16 hops, OSPF has no restriction.
- *Multipath Routing.* Allows you to use multiple paths of equal cost that connect the same points. You can then use these paths for load distribution that results in more efficient use of network bandwidth.
- *Area Routing.* Decreases the resources (memory and network bandwidth) consumed by the protocol and provides an additional level of routing protection.
- *Variable-Length Subnet Masks.* Allows you to break an IP address into variable-size subnets, conserving IP address space.
- *Routing Authentication.* Provides additional routing security.

OSPF supports the following physical network types:

- *Point-to-Point.* Networks that use a communication line to join a single pair of routers. A 56-Kbps serial line that connects two routers is an example of a point-to-point network.

- *Broadcast.* Networks that support more than two attached routers and are capable of addressing a single physical message to all attached routers. A token-ring network is an example of a broadcast network. Emulated LANs over ATM treat the ATM network as a broadcast network.
- *Non-Broadcast Multi-Access (NBMA).* Networks that support more than two attached routers but have no broadcast capabilities. An X.25 Public Data Network is an example of a non-broadcast network. For OSPF to function correctly, this network requires extra configuration information about other OSPF routers attached to the non-broadcast network. ATM Native treats the ATM interface as a Non-Broadcast Multiple Access (NBMA) interface.
- *Point-to-Multipoint.* Networks that support more than two attached routers, have no broadcast capabilities, and are non-fully meshed. A frame relay network without PVC between all the attached routers is an example of a Point-to-Multipoint network. Like non-broadcast networks, extra configuration information about other OSPF routers attached to the network is required.

Designated Router

Every broadcast or non-broadcast multi-access network has a designated router that performs two main functions for the routing protocol; it originates network link advertisements and it becomes adjacent to all other routers on the network.

When a designated router originates network link advertisements, it lists all the routers, including itself, currently attached to the network. The link ID for this advertisement is the IP interface address of the designated router. By using the subnet/network mask, the designated router obtains the IP network number.

The designated router becomes adjacent to all other routers and is tasked with synchronizing the link state databases on the broadcast network.

The OSPF Hello protocol elects the designated router after determining the router's priority from the *Rtr Pri* (router priority) field of the Hello packet. When a router's interface first becomes functional, it checks to see if the network currently has a designated router. If it does, it accepts that designated router regardless of that router's priority, otherwise, it declares itself the designated router. If the router declares itself the designated router at the same time that another router does, the router with higher router priority (*Rtr Pri*) becomes the designated router. If both router priorities are equal, the one with the higher router ID is elected.

Once the designated router is elected, it becomes the end-point for many adjacencies. On a broadcast network, this optimizes the flooding procedure by allowing the designated router to multicast its Link State Update packets to the address ALLSPFRouters (224.0.0.5) rather than sending separate packets over each adjacency.

Configuring OSPF

The following steps outline the tasks required to get the OSPF protocol up and running. The sections that follow explain each step in detail, including examples.

Before your router can run the OSPF protocol, you must:

1. Set the OSPF router ID. (See "Setting OSPF Router IDs" on page 1023.)

2. Define OSPF areas attached to the router. If no OSPF areas are defined, a single backbone area is assumed. (See “Defining Backbone and Attached OSPF Areas”.)
3. Define the router’s OSPF network interfaces. Set the cost of sending a packet out on each interface, along with a collection of the OSPF operating parameters. (See “Setting OSPF Interfaces” on page 1026.)
4. If the router interfaces to non-broadcast networks (X.25, Frame-Relay or ATM Native), set additional interface parameters. (See “Setting Non-Broadcast Network Interface Parameters” on page 1028 and “Configuring Wide Area Subnetworks” on page 1028.)
5. If you want the router to import routes learned from other routing protocols running on this router (for example, RIP), enable AS boundary routing. In addition, you must define whether routes are imported as Type 2 or Type 1 externals. (See “Enabling AS Boundary Routing” on page 1029.)

Setting OSPF Router IDs

Every router in an OSPF routing domain must be assigned a unique 32-bit router ID. Choose the value used for the OSPF router ID as follows:

- If you use the ROUTERID configuration statement, the value configured is used as the OSPF router ID. The value must be one of the stack’s configured interface IP addresses.
- If the ROUTERID configuration statement is not used, one of the OSPF interface addresses will be used as the OSPF router ID.

Defining Backbone and Attached OSPF Areas

Figure 49 on page 1024 shows a sample diagram of the structure of an OSPF routing domain. One division is between IP subnetworks within the OSPF domain and IP subnetworks external to the OSPF domain. The subnetworks included within the OSPF domain are subdivided into regions called *areas*. OSPF areas are collections of contiguous IP subnetworks. The function of areas is to reduce the OSPF overhead required to find routes to destinations in a different area. Overhead is reduced both because less information is exchanged between routers and because fewer CPU cycles are required for a less complex route table calculation.

Every OSPF routing domain must have at least a *backbone area*. The backbone is always identified by area number 0.0.0.0. For small OSPF networks, the backbone is the only area required. For larger 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 transit areas.

Routers that attach to more than one area function as area *border routers*. All area border routers are part of the backbone, so a border router must either attach directly to a backbone IP subnet or be connected to another backbone router over a virtual link. In addition, there must be a collection of backbone subnetworks and virtual links that connects all of the backbone routers.

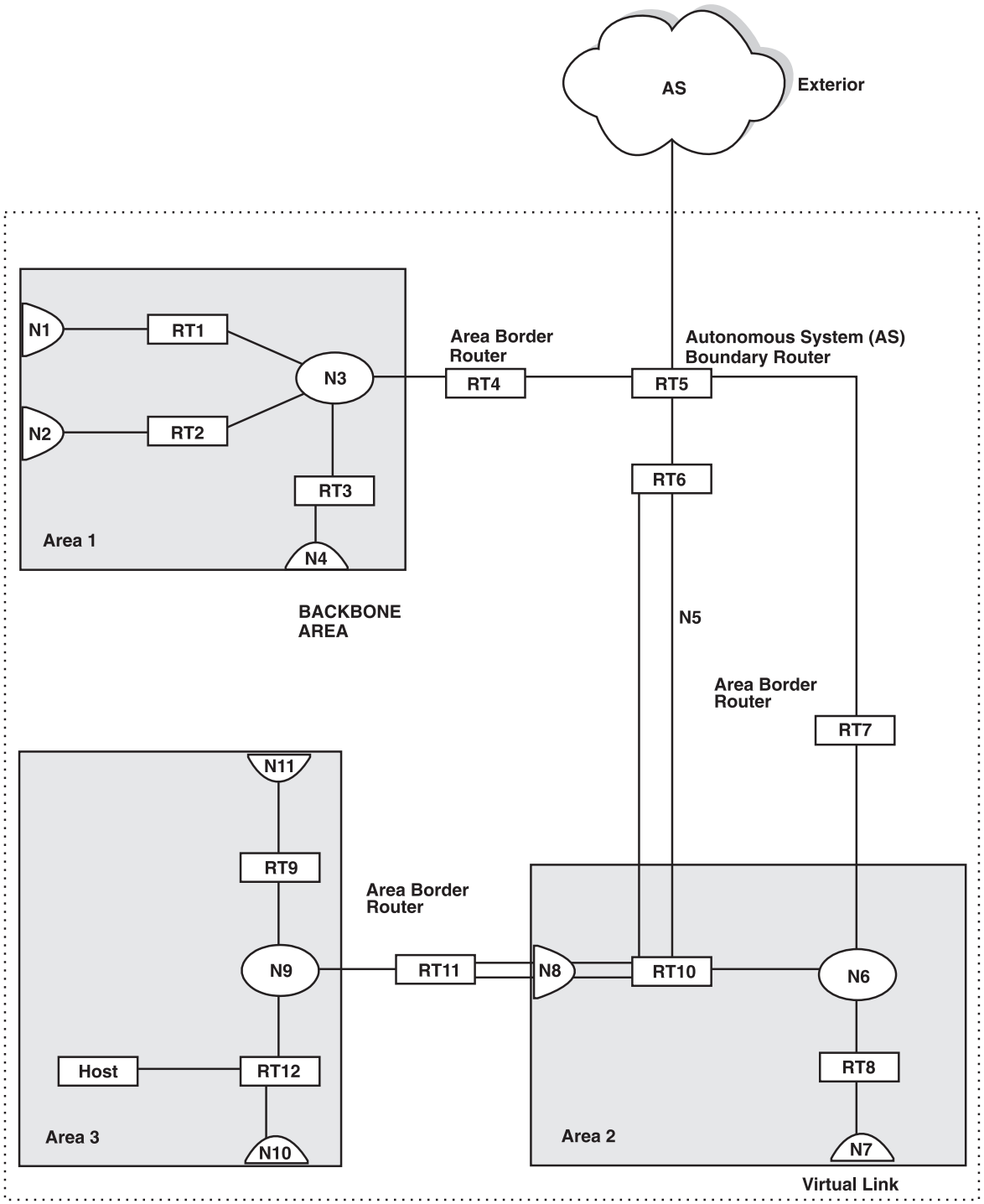


Figure 49. OSPF Areas

The information and algorithms used by OSPF to calculate routes vary according to whether the destination IP subnetwork is within the same area, in a different area within the same domain, or external to the OSPF domain. Every router maintains a complete map of all links within its area. All router to multi-access network, network to multi-access router, and router to router links are included in the map. A shortest path first algorithm is used to calculate the best routes to destinations within the area from this map. Routes between areas are calculated from summary

advertisements originated by area border routers for IP subnetworks, IP subnetwork ranges, and autonomous system external (ASE) boundary routers located in other areas of the OSPF domain. External routes are calculated from ASE advertisements that are originated by ASE boundary routers and flooded throughout the OSPF routing domain.

The backbone is responsible for distributing inter-area routing information. The backbone area consists of any of the following:

- Networks belonging to Area 0.0.0.0
- Routers attached to those networks
- Routers belonging to multiple areas
- Configured virtual links

Use the AREA configuration statement to define 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.

When area border routers are configured, parameters on the AREA and RANGE configuration statements can be used to control what OSPF route information crosses the area boundary.

One option is to use the AREA statement to define an area as a *stub*. OSPF ASE advertisements are never flooded into stub areas. In addition, the AREA statement has an option to suppress origination into the stub of summary advertisements for inter-area routes. Area border routers advertise default routes into stub areas. Traffic within the stub destined for unknown IP subnets is forwarded to the area border router. The border router uses its more complete routing information to forward the traffic on an appropriate path toward its destination. An area cannot be configured as a stub if it is used as a transit area for virtual links.

In summary, you can define an area as a stub when:
- There is no requirement for the area to handle transit backbone traffic.
- It is acceptable for area routers to use an area-border-router-generated default for traffic destined outside the AS.
- There is no requirement for area routers to be AS boundary routers (OSPF routers that advertise routes from external sources as AS external advertisements).

In this case, only the area border routers and backbone routers will have to calculate and maintain AS external routes.

The other option is to use IP subnet address ranges to limit the number of summary advertisements that are used for inter-area advertisements of an area's subnets. A range is defined by an IP address and an address mask. Subnets are considered to fall within the range if the subnet IP address and the range IP address match after the range mask has been applied to both addresses.

When a range is added for an area at an area border router, the border router suppresses summary advertisements for subnets in the areas that are included in 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 RANGE configuration statement.

Note that if the range is not advertised, there will be no inter-area routes for any destination that falls within the range. Also note that ranges cannot be used for areas that are used as transit areas by virtual links.

Setting OSPF Interfaces

OSPF interfaces are a subset of the IP interfaces defined to the TCP/IP stack. The parameters configured for OSPF interfaces determine the topology of the OSPF domain, the routes that will be chosen through the domain, and the characteristics of the interaction between directly connected OSPF routers. The OSPF_Interface configuration statement is used to define an OSPF interface and to specify its characteristics.

OSPF Domain Topology

The definition of the topology of an OSPF domain depends on a definition of which routers are directly connected across some physical media or subnetwork technology and the area to which those connections are a part. The basic case is for all routers attached to a physical subnetwork to be directly connected, but it is possible to define multiple IP subnetworks over a single physical subnetwork. In that case, OSPF will consider routers to be directly connected only when they have OSPF interfaces attached to the same IP subnetwork. It is also possible to have cases where routers attached to the same subnetwork do not have a direct link layer connection.

For LAN media, directly connected OSPF routers are determined from the IP subnetwork and physical media associated with an OSPF interface. The IP address, along with the subnet mask, defined with the OSPF_Interface configuration statement, determine the IP subnetwork to which the OSPF interface attaches. The *net index* associated with the IP interface determines the physical subnetwork to which the OSPF interface attaches. The broadcast capability of LANs allows OSPF to use multicast Hello messages to discover other routers that have interfaces attached to the same IP subnetwork.

LANs can be used to connect an OSPF router with IP hosts. In this case, it is still necessary to define an OSPF interface to any IP subnetwork that is defined for the LAN. Otherwise, OSPF will not generate routes with those IP subnetworks as destinations. To prevent OSPF Hello traffic on these LANs without other attached routers, the network can be defined as a non-broadcast multi-access network. The router priority should also be set to zero because no designated router is required.

The requirements for configuring OSPF interfaces that attach to serial lines vary with the lower layer technology.

For point-to-point lines, only one other router is accessible over the interface, so the directly connected router can be determined without additional configuration.

For subnetwork technologies like frame relay, ATM, and X.25 that support connections to multiple routers over a single serial line, the configuration of the OSPF interfaces is similar to that for a LAN, but because directly connected routers are not discovered dynamically for these subnetwork technologies, additional configuration is required to specify directly connected neighbors. For more information on the required configuration, see “Configuring Wide Area Subnetworks” on page 1028.

Costs for OSPF Links

OSPF calculates routes by finding the least-cost path to a destination. The cost of each path is the sum of the costs for the different links in the path. Table 44 shows sample costs for OSPF links.

Correctly configuring the costs according to the desirability of using interfaces for data traffic is critical for obtaining the desired routes through an OSPF domain. The factors that make individual links more or less desirable may vary in different networks, but the most common goal is to choose routes with the least delay and the most capacity. In general, this policy can be achieved by making the cost of a link inversely proportional to the bandwidth of the media used for the physical subnetwork.

A recommended approach is to use a cost of one for the highest bandwidth technology. For example, use the value 1 as the cost for an interface running 100 Mbps ATM.

Table 44. Sample Costs for OSPF Links

Interface Bandwidth	Cost
155 Mbps ATM	1
Ethernet	10
16 Mbps Token-Ring	6
4 Mbps Token-Ring	25
serial line	Cost based on bandwidth
Emulated Token-Ring (See note.)	1
Emulated Ethernet (See note.)	1

Note: An Emulated Token Ring or Ethernet will run at the interface speed (for example, 155 Mbps), and should be configured with a cost of 1.

ATM can attach to networks at a slower rate than the maximum line speed. For example, if the router has a port that is capable of 155 Mbps, and a router connects to it with 25 Mbps, that link will still be treated as a cost of 1. The OSPF weighting is on an interface basis.

Changing the Cost of OSPF Links

The cost of an OSPF interface can be dynamically changed using the MODIFY *procname*,OSPF,WEIGHT,NAME=*name*,COST=*cost* command, where name is the name of the interface and cost is the cost of using the interface. This new cost is flooded quickly throughout the OSPF routing domain, and modifies the routing immediately.

The cost of the interface will revert to its configured value whenever the router is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF interface in the configuration file.

Interactions between Neighbor Routers

A number of the values configured with the OSPF_Interface configuration statement are used to specify parameters that control the interaction of directly connected routers. They include:

- Retransmission interval
- Transmission delay
- Router priority

- Hello interval
- Dead router interval
- Demand Circuit
- Hello Suppression
- Poll Interval
- Authentication key

In most cases, the default values can be used.

Note: The Hello interval, the dead router interval, and the authentication key must have the same value for all OSPF routers that attach to the same IP subnetwork. If the values are not the same, routers will fail to form direct connections (adjacencies).

Setting Non-Broadcast Network Interface Parameters

If the router is connected to a non-broadcast, multi-access network, such as an X.25 PDN, you have to configure the following parameters to help the router discover its OSPF neighbors. This configuration is necessary only if the router will be eligible to become designated router of the non-broadcast network.

First configure the Non_Broadcast and NB_Poll_Interval parameters on the OSPF_Interface configuration statement.

Then configure the IP addresses of all other OSPF routers that will be attached to the non-broadcast network. For each router configured, you must also specify its eligibility to become the designated router. Use the DR_Neighbor parameter for neighbors that are eligible to become the designated router. Use the No_DR_Neighbor parameter for neighbors that are not eligible to become the designated router.

Setting non-broadcast can also be used to force a network without any other OSPF routers to be advertised. The router priority for the interface should be set to zero and no neighbors should be defined.

Configuring Wide Area Subnetworks

Frame Relay, ATM Native, and X.25 allow direct connections between multiple routers over a single serial line. Additional configuration is required for OSPF interfaces that attach to this kind of network. Because OSPF protocol messages are sent directly to specific neighbors on these networks, configuration is used instead of dynamic discovery to determine neighbor relationships and router roles.

Note: The configurations described in this section do not apply to point-to-point networks.

OSPF can assume either of two patterns for the direct connections between routers across these subnetworks:
- Point-to-Multipoint
- Non-broadcast multi-access (NBMA)

The key factor that distinguishes these two patterns is whether or not there is a direct connection between all pairs of routers that attach to the subnetwork (*full mesh connectivity*) or whether some of the routers are only connected through multi-hop paths with other routers as intermediates (*partial mesh connectivity*).

Non-broadcast multi-access (NBMA) requires *full mesh connectivity* while point-to-multipoint requires only *partial mesh connectivity*.

Point-to-multipoint is the default choice because it works for both full mesh connectivity and partial mesh connectivity. But when full mesh connectivity is available, NBMA is a more efficient solution.

Configuring Point-to-Multipoint Subnetworks

Point-to-multipoint can be configured more easily than NBMA because there are no designated routers, but neighbor relationships must be configured for all pairs of routers that will exchange data traffic directly across the point-to-multipoint subnet. Each pair of directly connected routers will exchange Hello messages, so one side can discover the other through these messages. The router configured to send the first Hello message, however, must have the IP address of its neighbor configured using the No_DR_Neighbor parameter on the OSPF_Interface statement.

It is important to remember that OSPF will not calculate the correct routes if some of the routers attached to a subnetwork represent it as NBMA and others represent it as point-to-multipoint. Therefore, never set Non_Broadcast=YES on the OSPF_Interface statement for any interface to a point-to-multipoint network.

Configuring NBMA Subnetworks

For NBMA IP subnetworks, some subset of the attached OSPF routers are configured to be eligible to be the designated router (DR). Each router eligible to be the DR periodically sends Hello messages to all other routers eligible to be the DR. These messages are used in the protocol to elect a DR and a backup DR. Both the DR and the backup DR periodically exchange Hello messages with all other OSPF routers that are attached to the NBMA IP subnetwork. Also, the flow of OSPF route information across the NBMA IP subnetwork is only between each of the attached routers and the DR or backup DR.

Select NBMA by setting Non_Broadcast=YES on the OSPF_Interface statement for interfaces that attach to an NBMA subnetwork. This must be used for all interfaces that attach to the NBMA network.

The configuration required for an OSPF router that attaches to an NBMA subnetwork depends on whether or not that router is eligible to become the DR.

- For a router not eligible to become a DR, the Router_Priority parameter on the OSPF_Interface statement must be set to 0.
- For a router eligible to become a DR, the OSPF_Interface statement must be used to set the router priority to a nonzero value and the DR_Neighbor and No_DR_Neighbor parameters must be used to identify all of the OSPF routers with interfaces attached to the NBMA subnetwork and to indicate which of them are eligible to become DR.

Note: In a star configuration, use the DR_Neighbor and No_DR_Neighbor parameters at the hub (neighbors at the remote site do not need to be configured).

Enabling AS Boundary Routing

To import routes learned from other protocols (for example, RIP) into the OSPF domain, use the AS_Boundary_Routing configuration statement. You must do this even if the only route you want to import is the default route (destination 0.0.0.0).

When enabling AS boundary routing, you must specify which external routes you want to import. You can choose to import, or not to import, routes belonging to the following categories:

- RIP routes
- Static routes
- Direct routes

For example, you could choose to import direct routes, but not RIP or static routes.

Independent of the above external categories, you can also configure whether or not to import subnet routes into the OSPF domain. This configuration item defaults to ENABLED (subnets are imported).

The metric type used in importing routes determines how the imported cost is viewed by the OSPF domain. When comparing two type 2 metrics, only the external cost is considered in picking the best route. When comparing two type 1 metrics, the external and internal costs of the route are combined before making the comparison.

Configuring OSPF over ATM

The options for configuring OSPF over an ATM subnetwork depend on whether LAN Emulation or ATM Native is being used for the IP layer. In the case of LAN Emulation, OSPF is configured in the same way as for a real LAN. For ATM Native, the OSPF configuration options are the same as for Wide Area Subnetworks. See “Configuring Wide Area Subnetworks” on page 1028. Both NBMA and Point-to-Multipoint configurations are supported.

Configuring OSPF over ATM Native

OSPF over ATM Native requires the following configuration steps:

1. Use the OSPF_Interface statement for the ATM interface. Set the OSPF parameters including Designated-Router(DR) eligibility.
2. Set Non_Broadcast=YES on the OSPF_Interface statement for the ATM interface. This also needs to be set on all interfaces on every router that is connected to an ATM Native Logical IP subnet (LIS).
3. Use the DR_Neighbor and No_DR_Neighbor parameters of the OSPF_Interface statement to define the other routers on the Logical IP Subnet (LIS) that you wish to share OSPF routing information with.

   Note: All routers that are eligible to be Designated Routers (DRs) need to be configured with the neighbor information. Only one router in every LIS needs to be DR; however, if other routers are also configured to be DR-eligible, the LIS is more capable of recovering when an outage occurs.

Other Configuration Tasks

Setting Virtual Links

To maintain backbone connectivity, you must have all of your backbone routers interconnected either by permanent or virtual links. You can configure virtual links between any two area border routers that share a common non-backbone and non-stub area. Virtual links are considered to be separate router interfaces

connecting to the backbone area. Therefore, you are asked to also specify many of the interface parameters when configuring a virtual link.

Virtual links must be configured in each of the link's two end-points. Note that you must enter OSPF router IDs in the same form as IP addresses.

No cost is configured for a virtual link because the cost is the OSPF intra-area cost between the virtual link end-points through the transit area.

Configuring for Routing Protocol Comparisons

If you use a routing protocol in addition to OSPF, or when you change your routing protocol to OSPF, you must set the Routing Protocol Comparison.

OSPF routing in an AS occurs on these three levels: intra-area, inter-area, and exterior.

Intra-area routing occurs when a packet's source and destination address reside in the same area. Information that is about other areas does not affect this type of routing.

Inter-area routing occurs when the packet's source and destination addresses reside in different areas of the same AS. OSPF does inter-area routing by dividing the path into three contiguous pieces: an intra-area path from source to an area border router; a backbone path between the source and destination areas; and then another intra-area path to the destination. You can visualize this high-level of routing as a star topology with the backbone as hub and each of the areas as a spoke.

Exterior routes are paths to networks that lie outside the AS. These routes originate either from routing protocols, such as RIP, or from static routes entered by the network administrator. The exterior routing information provided by RIP does not interfere with the internal routing information provided by the OSPF protocol.

AS boundary routers can import exterior routes into the OSPF routing domain. OSPF represents these routes as AS external link advertisements.

OSPF imports external routes in separate levels. The first level, called type 1 routes, is used when the external metric is comparable to the OSPF metric (for example, they might both use delay in milliseconds). The second level, called external type 2 routes, assumes that the external cost is greater than the cost of any internal OSPF (link-state) path.

Imported external routes are tagged with 32 bits of information. In a router, this 32-bit field indicates the AS number from which the route was received. This enables more intelligent behavior when determining whether to re-advertise the external information to other autonomous systems.

OSPF has a 4-level routing hierarchy (see Figure 50 on page 1032). The COMPARISON configuration statement tells the router where the RIP/static routes fit in the OSPF hierarchy. The two lower levels consist of the OSPF internal routes. OSPF intra-area and inter-area routes take precedence over information obtained from any other sources, all of which are located on a single level.

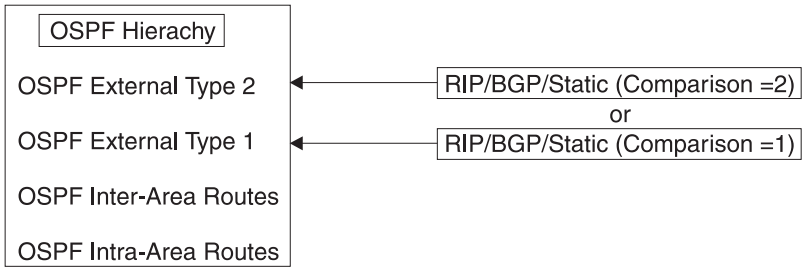


Figure 50. OSPF Routing Hierarchy

To put the RIP/static routes on the same level as OSPF external type 1 routes, set the comparison to 1. To put the RIP/static routes on the same level as OSPF external type 2 routes, set the comparison to 2. The default setting is 2.

For example, suppose the comparison is set to 2. In this case, when RIP routes are imported into the OSPF domain, they will be imported as type 2 externals. All OSPF external type 1 routes override received RIP routes, regardless of metric. However, if the RIP routes have a smaller cost, the RIP routes override OSPF external type 2 routes. The comparison values for all of your OSPF routers must match. If the comparison values set for the routers are inconsistent, your routing will not function correctly.

Demand Circuit

Demand circuits are network segments whose costs vary with usage; charges can be based both on connect time and on bytes or packets transmitted. Examples of demand circuits include ISDN circuits, X.25 SVCs, and dial-up lines. The periodic nature of OSPF routing traffic requires a link's underlying data-link connection to be constantly open, resulting in unwanted usage charges on demand circuits. Configuring an interface as a demand circuit results in the suppression of OSPF Hellos and OSPF routing information refreshes, allowing the underlying data-link connection to be closed when not carrying application traffic.

Demand circuits and regular network segments (for example, leased lines) are allowed to be combined in any manner; there are no topological restrictions on the demand circuit support. However, while any OSPF network segment can be defined as a demand circuit, only point-to-point networks receive the full benefit. When broadcast and NBMA networks are declared demand circuits, routing update traffic is reduced but the periodic sending of Hellos is not, which still requires that the data-link connections remain constantly open.

A demand circuit can be configured for any interface. There is no dependence on physical media or the model used by OSPF for the route calculation. When the demand circuit is configured and there are no compatibility problems:

- Only Link State Advertisements (LSAs) with real changes will be advertised over the interface. Normally, OSPF's reliable flooding algorithm causes LSAs to be refreshed with a new instance every 30 minutes even if topology changes have occurred.
- The DoNotAge bit will be set for LSAs flooded over the interface. This is required since they will not be refreshed over the interface.

Request Hello Suppression

This is an additional parameter that you can use to configure an interface to request Hello suppression. This parameter will have value for point-to-point and point-to-multipoint interfaces.

PP_Poll_Interval

If Demand Circuit and Hello Suppression are configured for an interface, the PP_Poll_Interval parameter of the OSPF_Interface statement will be used by OSPF to try to reestablish a connection when a point-to-point line is down because there was a failure to transmit data but the network still appears to be operational.

Converting from RIP to OSPF

To convert your Autonomous System from RIP to OSPF, install OSPF one router at a time, leaving RIP running. Gradually, all your internal routes will shift from being learned via RIP to being learned by OSPF (OSPF routes have precedence over RIP routes). If you want to have your routes look exactly as they did under RIP (in order to check that the conversion is working correctly) use hop count as your OSPF metric. Do this by setting the cost of each OSPF interface to 1.

After installing OSPF on your routers, turn on AS boundary routing in all those routers that still need to learn routes via other protocols (BGP, RIP, and statically configured routes). The number of these AS boundary routers should be kept to a minimum.

Finally, you can disable the receiving of RIP information on all those routers that are not AS boundary routers.

Chapter 28. Configuring the NCPROUTE Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

NCPROUTE is a server that provides an alternative to using the Network Control Program (NCP) as a static host-independent IP router. NCPROUTE has the following effects:

- NCP becomes an active RIP router on a TCP/IP network
- NCP becomes responsive to SNMP route table queries

Notes:

1. NCPROUTE requires NCP V7R1, or later.
2. NCPROUTE requires SNALINK LU0 when using NCP V7R3 or previous.
3. SNALINK and IP over CDLC is supported for ESCON, BCCA, and CADS channels.
4. IP over CDLC can be used instead of SNALINK when using NCP V7R4, or later.
5. If using RIP Version 2, NCPROUTE requires NCP V7R6, or later. Also, the NCP generation definition must have VSUBNETS=YES specified on the BUILD statement.
6. NCP versions V6R1 and V6R2 only support static IP routing. NCP uses these static route tables to deliver datagrams over connected TCP/IP networks. NCP V7R1 can be specified only as a host-dependent router and it requires the NCPROUTE server to function as a RIP router.
7. If using NCPROUTE with SNALINK, IP over CDLC channels and OROUTED, you should customize the NCST interface metric on the NCP client side for the SNALINK NCST connection so the routes will be less preferred. This will cause RouteD to prefer routes from the IP over CDLC interface over the ones from the SNALINK interface. To customize the interface metric, see the *interface metric* option in “Step 8: Configure the NCPROUTE Gateways Data Set (Optional)” on page 1051. Do the same for the SNALINK interface on the MVS host side by customizing the metric in the BSDROUTINGPARMS statement. RIP traffic will be carried over the IP over CDLC interface, while transport PDUs (for example, Hello, Add Route Request, Delete Route Request) will be carried over the SNALINK interface.
8. NCPROUTE does not support zero subnets.

Understanding NCPROUTE

NCPROUTE provides dynamic route table updates for one or more NCP clients that have been generated as IP routers and have NCPROUTE specified as the NCPROUTE server. NCPROUTE tables are updated periodically in the NCP client based on updates sent by the NCPROUTE server. These updates reflect dynamic changes in route states.

An NCPROUTE server at the host uses the Routing Information Protocol (RIP), described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). The same routing protocols are used by the OROUTED server. NCPROUTE is implemented as a RIP server operating on an MVS host connected to a RIP client in the NCP.

Together they provide the appearance to the TCP/IP network of an IP router using the RIP protocol. The same client/server pair also provides SNMP agent support for network management route table queries. For a brief description of RIP (Versions 1 and 2), see “Routing Information Protocol (RIP)” on page 918.

RIP Versions 1 and 2 are currently supported by NCPROUTE. For a brief description of RIP Version 2 features, see “RIP Version 2” on page 919.

Environment

The NCPROUTE server:

- Supports multiple host-attached, link-attached, and remote link-attached NCP clients as illustrated in Figure 51
- Generates RIP datagrams for the NCP to send
- Maintains separate routing tables for each NCP client
- Generates SNMP route table responses for each NCP SNMP agent

The client NCP unit appears as an active router to other RIP routers on the network. Multiple NCP clients can connect to the same NCPROUTE server. Each NCP will appear as an IP router to the rest of the network. Each NCP client must have one or more LU0 sessions established with SNALINK. One LU0 session per client is used as the primary session, with the remaining sessions serving as backups.

Figure 51 illustrates the different ways the NCPROUTE server can support NCP clients. NCP3 and NCP4 are host-attached NCP clients, NCP5 and NCP6 are link-attached NCP clients, and NCP1 and NCP2 are remote link-attached NCP clients.

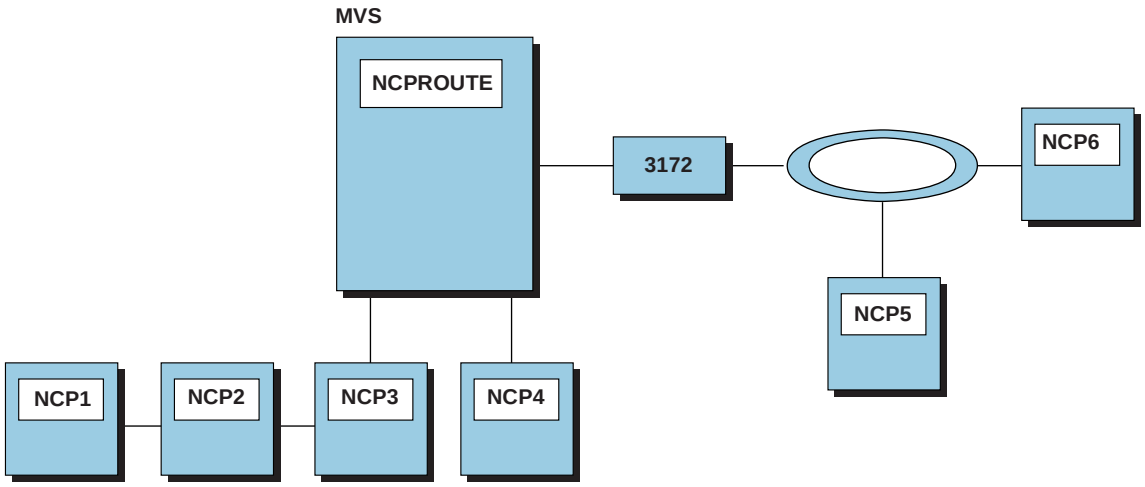


Figure 51. NCPROUTE Environment

Server Requirements

NCPROUTE processes RIP and SNMP datagrams addressed to all attached NCP units, generates datagrams for the NCP units, and maintains the state of each NCP unit's routing tables.

SNMP support is limited to route table queries. Queries are made to the NCP, which sends the request to the NCPROUTE server for processing.

NCPROUTE Operation

An NCP's IPOWNER statement defines the controlling host and the interface this NCP client must use to reach the host. The NCP client initiates contact with NCPROUTE by sending a datagram, known as a "Hello" message, to the controlling host. It transmits this datagram on UDP port 580.

Note: The port number is generated in the NCP (using the UDPPORT keyword on the IPOWNER statement) and configured in NCPROUTE.

The "Hello" message identifies the client NCP and determines which member from the *hlq*.NCPROUTE.GATEWAYS partitioned data set to use for this NCP's route table. Any valid MVS data set name can be used for the gateways data set.

The NCP client then sends a list of its inactive links to NCPROUTE. NCPROUTE uses additional routes defined for this NCP in the NCPROUTE gateways data set, as defined in the NCPROUTE profile. It also uses the inactive links provided dynamically by the NCP to build the current route table for this NCP. The following process is repeated for each NCP that has been generated to act as a RIP router:

- A RIP packet arrives at the NCP client from a foreign router
- The NCP client sends this datagram to the NCPROUTE server
- The NCPROUTE server processes the RIP packet
- The NCPROUTE server creates a RIP update for an NCP client
- This update is sent to the NCP client
- The NCP client transmits the datagram to the network

NCPROUTE sends route table updates to each NCP client every 30 seconds. After a client has been activated, updates must be supplied over each of its interfaces every 30 seconds. The NCPROUTE server creates these updates and sends them to the NCP client along with the IP addresses of other RIP routers that the NCP client should send them to.

At the same time, adjacent RIP routers are providing periodic updates every 30 seconds to NCPROUTE. These updates are sent by the NCP client to the NCPROUTE server, where they are processed, and the results are reflected in future updates back to the NCP client.

The NCP client sends all SNMP and RIP datagrams to the NCPROUTE server for processing. The NCPROUTE server provides RIP packets and SNMP replies to the NCP client to send to their final destination.

NCPROUTE Gateways

Passive RIP Route: Information about passive routes is put in NCP's and NCPROUTE's routing tables. A passive entry in NCPROUTE's routing table is used as a placeholder to prevent a route from being propagated and from being

overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router.

Using passive routes can create routing loops, so care must be taken when creating them.

Do not define passive routes such as these:
- A to C is via B
- B to C is via A

Passive routes should be used when adding routes where the host or network is not running RIP. Passive routes should also be used when adding a default route, because this is the only way to prevent a route from timing out.

External RIP Route: External routes are managed by other protocols, for example, the External Gateway Protocol (EGP). NCPROUTE needs to know not to interfere with these routes and not to delete them.

An external entry exists in the NCPROUTE routing table as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. NCPROUTE does not manage an external route. Therefore, NCPROUTE only knows that there is an existing route to the host or network and that is the route known by NCP.

External routes should be used when the local machine is running a non-RIP routing protocol that dynamically changes the TCP/IP routing tables. The remote machine does not need to run any routing protocol, since the only concerns are how to route traffic from the local machine to the remote machine, and how to prevent multiple routing protocols from interfering with each other.

NCPROUTE Active Gateways

Active gateways are treated as remote network interfaces. Active gateways are routers that are running RIP, but are reached through a medium that does not allow broadcasting or multicasting and is not point-to-point, for example, Hyperchannel. NCPROUTE normally requires that routers be reachable by broadcast or multicast for non-point-to-point links or by unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to NCPROUTE's interface list. NCPROUTE will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non broadcast- capable, non multicast-capable, and non point-to-point network, and is directly connected to the local host.

NCPROUTE will communicate with active routes by unicast transmissions to the gateway address. Routes are not added immediately to either NCPROUTE or the NCP routing table. They are added and propagated normally when route advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast communication on non-point-to-point links. Interfaces that are not broadcast, not point-to-point, and are not active gateways are assumed to be loopback interfaces to the local machine. Also, while a route to an active gateway might timeout, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

NCPROUTE Gateways Summary

Table 45 provides a list of NCPROUTE gateways and their characteristics.

Table 45. NCPROUTE Gateways Summary

	Propagate	Kernel	NCPROUTE	Timeout
Dynamic (5)	Yes	Yes	Yes	Yes
Passive	No (6)	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes

5 Dynamic routing is provided by NCPROUTE.

6 Except directly-connected passive routes. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in a NCP client.

RIP Input/Output Filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by NCPROUTE and consist of:

- Route receiving (unconditional and conditional)
- Route noreceiving
- Route forwarding (unconditional and conditional)
- Route noforwarding
- Interface Supply switch
- Interface RIP On/Off switch
- Gateway noreceiving

For more information on these filters, see “Step 8: Configure the NCPROUTE Gateways Data Set (Optional)” on page 1051.

Configuration Process

Steps to Configure NCPROUTE:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP.
2. If using SNALINK, configure VTAM and SNALINK applications.
3. If using IP over CDLC, configure IP over CDLC DEVICE and LINK statements.
4. Update the NCPROUTE cataloged procedure.
5. Update *hlq*.ETC.SERVICES.
6. Configure the host-dependent NCP clients.
7. Configure the NCPROUTE profile data set for SNMP support and specification of an NCPROUTE gateways data set (optional).
8. Configure the NCPROUTE gateways data set for each NCP client (optional).
9. If RouteD is not used, define a directly-connected host route to each NCP client.

The following diagram shows the network addresses used in the configuration examples:

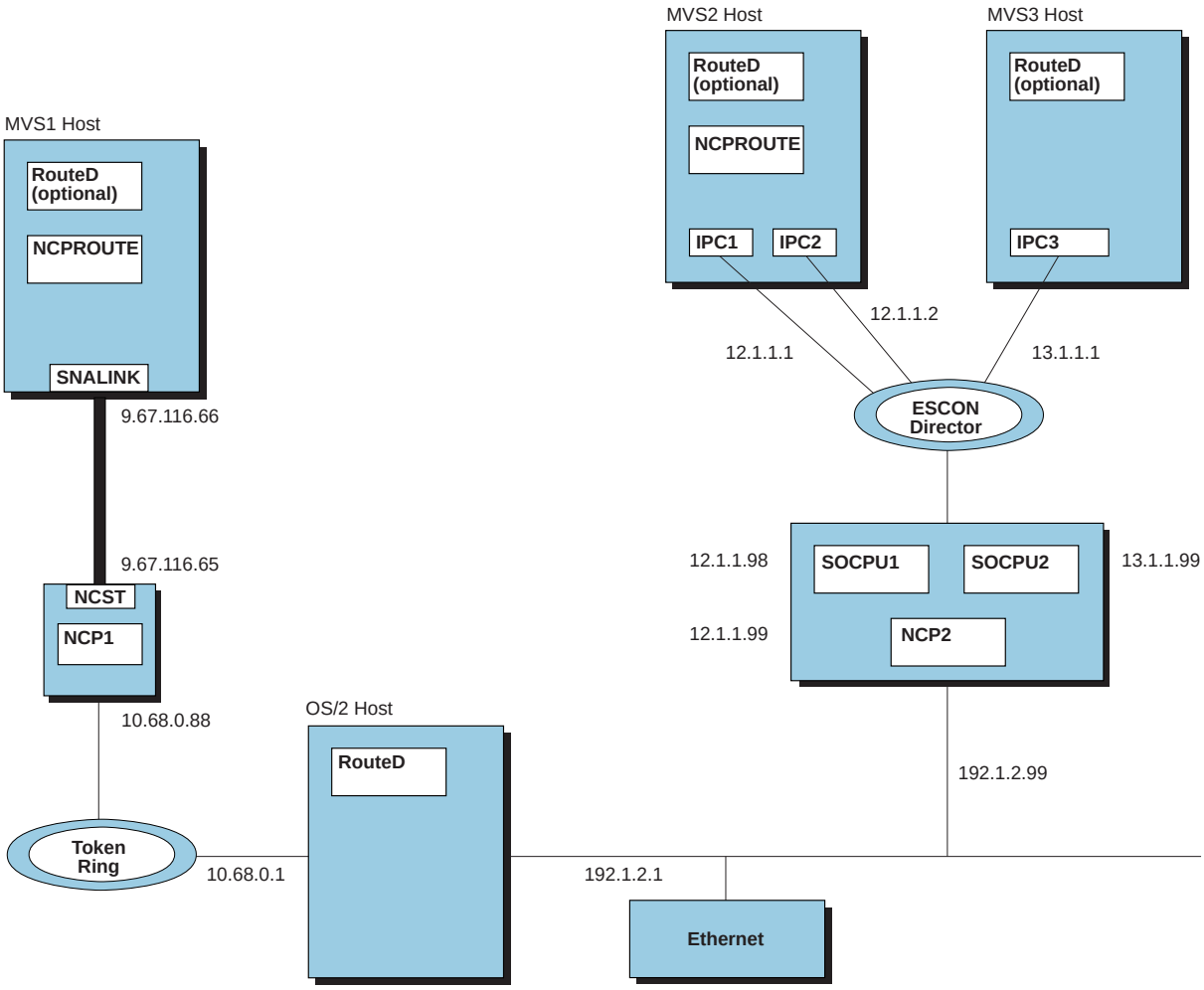


Figure 52. NCPROUTE Example Configuration

Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP

To have the NCPROUTE server started automatically when the TCPIP address space is started, include the name of the member containing the NCPROUTE cataloged procedure in the AUTOLOG statement in *hlq*.PROFILE.TCPIP:

```
AUTOLOG
   NCPROUT
ENDAUTOLOG
```

To ensure that UDP port 580 is reserved for the NCPROUTE server, also add the name of the member containing the NCPROUTE cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
   580 UDP NCPROUT
```

Note: This port number must match the one defined in the NCP generation definition (using the UDPPORT keyword on the IPOWNER statement) and assigned in *hlq*.ETC.SERVICES.

NCPROUTE also requires HOME and BSDROUTINGPARMS statements for the SNALINK type LU0 and IP over CDLC connections. For example, you would use

this HOME and BSDROUTINGPARMS statements and, optionally, GATEWAY statement for the configuration shown in Figure 52 on page 1040:

```
MVS1:   HOME
           9.67.116.66  SNALINK
           BSDROUTINGPARMS false
             SNALINK 2000     0    255.255.240.0    9.67.116.65
           ENDBSDROUTINGPARMS
MVS2:   HOME
           12.1.1.1   IPC1
           12.1.1.2   IPC2
           BSDROUTINGPARMS false
             IPC1    1000     0    255.255.255.0   12.1.1.98
             IPC2    1000     0    255.255.255.0   12.1.1.99
           ENDBSDROUTINGPARMS
MVS3:   HOME
           13.1.1.1   IPC3
           BSDROUTINGPARMS false
             IPC3    1000     0    255.255.255.128   13.1.1.99
           ENDBSDROUTINGPARMS
```

Notes:

1. If you are not using RouteD to manage the host routes, configure static routes to the NCP client or clients in the GATEWAY statement in *hlq*.PROFILE.TCPIP. If using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS is required to route Transport PDUs prior to OMPROUTE activation. Since the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both BSDROUTINGPARMS statement and the OMPROUTE configuration file. See “Step 9: Define a Directly Connected Host Route for the NCST Session” on page 1061 for sample definition. For more information on the GATEWAY statement, see “GATEWAY Statement” on page 193 to each NCP client.
2. A BSDROUTINGPARMS statement is required even though RouteD is not used.

You can find for a complete explanation of these configuration statements in *OS/390 SecureWay Communications Server: IP Configuration*.

Step 2: Configure VTAM and SNALINK Applications

If you are using NCP V7R3 or previous, NCPROUTE requires SNALINK type LU0 to run. For NCP V7R4, or later, IP over CDLC can be used instead of SNALINK. If SNALINK is to be used, verify that you have configured the SNALINK LU0 interface, defined it to VTAM with a VTAM APPL definition, and included the correct DEVICE and LINK statements in *hlq*.PROFILE.TCPIP.

If you are using the Cross Domain Resource (CSRSC), verify that the cross-domain resource managers are configured in VTAM.

Following is an example of an appropriate VTAM APPL definition:

```
*********************************************************************
*       SNALINK VTAM APPL DEFINITION                                *
*********************************************************************
SNALKLU1 APPL AUTH=(ACQ,VPACE),ACBNAME=SNALKLU1,EAS=12,PARSESS=YES,   *
              SONSCIP=YES,VPACING=0,SRBEXIT=YES
```

Note: The application name (the ACBNAME value, SNALKLU1, in this example) must match the REMLU interface definition in the NCP clients generation program. See the example in “Step 6: Configure the Host-Dependent NCP Clients” on page 1045 for more information.

Following is an example of corresponding DEVICE and LINK statements:

```
;
;  DEVICE AND LINK DEFINITIONS FOR SNALINK LU0
;
DEVICE SNA1LINK SNAIUCV SNALINK A04TOLU1 SNALPROC
LINK SNALINK SAMEHOST 1 SNA1LINK
;
```

Note: The LU name on the DEVICE statement (A04TOLU1 in this example) must match the LU name of the NCST interface definition in the NCP clients generation program. See the example in “Step 6: Configure the Host-Dependent NCP Clients” on page 1045 for more information.

If you want the SNALINK device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP. For example, `START SNA1LINK`. Otherwise, you will have to start the device manually.

Step 3: Configure the IP over CDLC DEVICE and LINK Statements

For NCPROUTE, IP over CDLC can be configured along with SNALINK for NCP V7R3, or later, or it can be used to replace SNALINK for NCP V7R4, or later.

Following is an example of corresponding DEVICE and LINK statements for the configuration shown in Figure 52 on page 1040 for the MVS2 host:

```
;
;  DEVICE AND LINK DEFINITIONS FOR IP OVER CDLC
;
DEVICE IPC1NCP CDLC 013 40 40 1024 1024
LINK IPC1 CDLC 0 IPC1NCP
;
DEVICE IPC2NCP CDLC 014 40 40 1024 1024
LINK IPC2 CDLC 0 IPC2NCP
```

Note: If you want a CDLC device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP, for example, START IPC1NCP. Otherwise, you will have to start the device manually.

Step 4: Update the NCPROUTE Cataloged Procedure

Update the NCPROUTE cataloged procedure by copying the sample in *hlq*.SEZAINST(NCPROUT) to your system or recognized PROCLIB. Specify NCPROUTE parameters and change the data set names to suit your local configuration. See Figure 53 on page 1051 for an illustration of NCPROUTE data set relationships.

NCPROUTE Cataloged Procedure (NCPROUT)

```
//NCPROUT PROC MODULE=NCPROUTE,PARMS='/'
//*
//*  TCP/IP for MVS
//*  SMP/E Distribution Name: EZBNRJCL
//*
//*         Licensed Materials - Program Property of IBM.
//*         This product contains "Restricted Materials of IBM"
//*         5655-HAL (C) Copyright IBM Corp. 1994
//*         All rights reserved.
//*         US Government Users Restricted Rights -
//*         Use, duplication or disclosure restricted
//*         by GSA ADP Schedule Contract with IBM Corp.
```

```
//*         See IBM Copyright Instructions
//*
//NCPROUT  EXEC PGM=&MODULE,
//            PARM='&PARMS',
//            REGION=4096K,TIME=1440
//*
//*  STEPLIB contains libraries to be accessed by NCPROUT.  Required
//*  libraries are the TCPIP executable module library and the NCP
//*  load library which contains a client's NCP load module and its
//*  Routing Information Table (RIT).
//*
//*  The C runtime libraries should be in the system's link list or add
//*  them to the STEPLIB definition here.  If you add them to STEPLIB,
//*  they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*        DD DSN=ncp.v7r1.ncpload,DISP=SHR
//*
//*  SYSPRINT contains output from NCPROUTE plus any enabled tracing.
//*  It can be a data set or SYSOUT.
//*
//SYSPRINT DD SYSOUT=*
//*
//*  SYSERR contains abnormal run-time error messages from NCPROUTE.
//*  It can be a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*  SYSMDUMP contains ABEND dump areas in case NCPROUTE abends.  The
//*  dump can be analyzed using IPCS for diagnosis.  It can be a data
//*  set or SYSOUT.
//*
//SYSMDUMP DD SYSOUT=*
//*
//*  NCPRPROF contains profile configuration information such as SNMP
//*  agent specifications and the name of a GATEWAYS partitioned data
//*  set (PDS).  A data set member can be optionally created for each
//*  NCP client and it can contain gateway definitions and NCPROUTE
//*  server options such as tracing and broadcasting of route tables.
//*
//*  The data set can be any sequential data set or a member of a
//*  partitioned data set (PDS).  For a sequential data set, specify
//*  FREE=CLOSE parameter for dynamic allocation support.
//*
//NCPRPROF DD DSN=TCPIP.SEZAINST(EZBNRPRF),DISP=SHR
//*
//*  SYSTCPD explicitly identifies which data set is to be used to
//*  obtain the parameters defined by TCPIP.DATA.  The SYSTCPD DD
//*  statement should be placed in the TSO logon procedure or in the
//*  JCL of any client or server executed as a background task.
//*  The data set can be any sequential data set or a member of a
//*  partitioned data set (PDS).
//*
//*  For more information please see "Understanding TCP/IP Data
//*  Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//*
//*  SERVICES points to an optional etc.services data set which can be
//*  used to override well-known ports in the ETC.SERVICES.
//*
//*SERVICES DD DSN=TCPIP.SEZAINST(EZAEB02J),DISP=SHR
//*
//*  MSNCPROU contains NCPROUTE's optional message repository for NLS
//*  support.
//*
//*MSNCPROU DD DSN=TCPIP.SEZAINST(EZBNRMSG),DISP=SHR,FREE=CLOSE
```

Specifying the NCPROUTE Parameters

The system parameters required by NCPROUTE are passed by the PARM parameter on the EXEC statement of the NCPROUTE cataloged procedure. Add your parameters to `PARMS='/'` in the PROC statement of the NCPROUTE cataloged procedure, making certain that:

- A slash (/) precedes the first parameter
- Each parameter is separated by a blank
- Mixed case is allowed for the parameters
- Blanks and comments are supported in the gateways data set. Comments are identified by a semicolon (**;**).

For example: `//NCPROUT PROC MODULE=NCPROUT,PARMS='/-s -t -t'`

Note: These parameters are also valid when starting the NCPROUTE server with the START command or when modifying NCPROUTE with the MODIFY command. For further information on parameters used with the MODIFY command, see "Controlling the NCPROUTE Address Space with the MODIFY Command" on page 1061.

-dp
Trace packets coming in and out of NCPROUTE for all NCP clients. The packets are displayed in data format.

-h Include host routes in the RIP responses. Adjacent routers a NCP client must be able to receive host routes; otherwise, NETWORK UNREACHABLE problems will occur.

-s Supply routing information for all NCP clients and override the supply settings in the NCP clients' gateways data sets.

-sl Supply local (directly connected) routes only for all NCP clients. This option is provided as a RIP output filter.

-sq
Suppress supplying routing information to all NCP clients and override the supply settings in the NCP clients' gateways data set.

-t Activate global tracing of actions for all NCP clients.

-tq
Deactivate tracing at all levels. This parameter suppresses tracing for all NCP clients and overrides the trace settings in the NCP clients gateway data set.

-t -t
Activate global tracing of packets for all NCP clients.

Note: There are no third or fourth level global tracing options like those in the NCPROUTE gateways data set members, however, additional levels can be specified using the MODIFY command for a specific NCP client. In any case, the system will use the highest of all the settings.

For more information, see Table 47 on page 1063.

All traces go to a standard output referred to by the //SYSPRINT DD statement in the NCPROUTE cataloged procedure. All abnormal runtime error messages go to the data set specified by the //SYSERR DD statement in the NCPROUTE cataloged procedure.

Step 5: Update hlq.ETC.SERVICES

NCPROUTE uses the *hlq*.ETC.SERVICES data set to determine the port number on which to run. This data set can be used to define a port number other than the reserved well-known port for NCPROUTE. This data set must exist for NCPROUTE to run.

The ETC.SERVICES is dynamically allocated using the standard search sequence for data set names. This data set also can be explicitly allocated in the NCPROUTE cataloged procedure using the //SERVICES DD statement.

The entries in *hlq*.ETC.SERVICES are case and column sensitive. They must be in lower case and begin in column 1.

Add the following lines to the *hlq*.ETC.SERVICES data set:

```
ncproute  580/udp
router    520/udp
```

Note: Verify that the NCPROUTE service port number is the port being used by the NCP clients. This number should match the port number defined in the NCP generation definition using the UDPPORT keyword on the IPOWNER statement. This port number does not necessarily have to match the reserved port number for NCPROUTE on the PORT statement in *hlq*.PROFILE.TCPIP.

The reserved router service port number is 520. It is required for the NCPROUTE transport of RIP packets to NCP clients which are responsible for broadcasting the packets to other RIP routers. It cannot be overridden.

If you want to use name aliases, refer to INFO APAR II08205 for information.

Step 6: Configure the Host-Dependent NCP Clients

You should refer to the appropriate NCP documentation for more information about defining and generating the NCP and creating route information tables.

- For more information about defining IP, refer to *NCP, SSP, and EP Resource Definition Guide*.
- For more information about the IP Dynamics function, refer to *NCP and EP Reference*.
- For more information about NCP generation definitions for IP, refer to *NCP, SSP, and EP Resource Definition Reference*.
- For more information about generating NCP as an IP router, refer to *NCP, SSP, and EP Generation and Loading Guide*.

Note: See notes in “Chapter 28. Configuring the NCPROUTE Server” on page 1035.

Generating the Routing Information Tables

To support IP dynamics, NCP’s Network Definition Facility (NDF) builds a routing information table (RIT) for networks and subnetworks for use by TCP/IP at NCP generation time.

The RIT consists of routing tables that are generated from the NCP IPROUTE and IPLOCAL statements. During NCP generation, the RIT is added as a member of the NCP load library partitioned data set ncp.v7r1.ncpload. You identify the member name of ncp.v7r1.ncpload that NCPROUTE uses at execution time with the NEWNAME parameter of the BUILD statement for each NCP client generation.

Determining the Gateway Route Table Name

There is one RIT in the ncp.v7r1.ncpload data set for each NCP client this server supports. The NCPROUTE server receives the NCP name from an NCP client in the “Hello” message. This name is used as the base to determine the member name in the ncp.v7r1.ncpload partitioned data set to use for the initial RIT for this NCP client. The RIT member name in the ncp.v7r1.ncpload data set is the NEWNAME parameter of the BUILD statement for the NCP generation with a suffix of **P** added. Specify a unique name on the NEWNAME parameter of the BUILD statement for each NCP client. This name is also used as the member name if the optional gateways data set (GATEWAYS_PDS) is specified in the NCPROUTE profile. The RIT is accessed by NCPROUTE from a //STEPLIB DD statement in the NCPROUTE cataloged procedure, LINKLST, or authorized library.

NCST Session Interface Definition

The NCP Connectionless SNA Transport (NCST) interface is used to establish a session that can provide a connection to another IP node (NCP or S/370) over a SNA network. Use this definition when using NCST PU interfaces to communicate with NCPROUTE using SNALINK devices with the MVS host. The NCST interface must be defined to match the SNALINK LU0 interface in VTAM so that an NCP client can establish connection with NCPROUTE. The LU statement in the NCST interface definition tells VTAM which interface to use for the SNALINK application. The following are important keywords in this definition:

NCST
Specifies the protocol type. Must be coded as IP for internet protocol.

INTFACE
Species the name of the interface and the maximum transfer unit (MTU) size for the NCST session to the VTAM owner (IPOWNER).

REMLU
Specifies the name of the remote LU for the SNALINK LU0 VTAM connection. This name must match:
- The APPLID in the PROC statement of the SNALINK cataloged procedure
- The application name in the VTAM APPL definition

Note: If you define a backup NCST SNALINK session, the REMLU can specify the primary logical name fo the remote LU or a different remote LU. Ensure that the MTU sizes are the same for the backup NCST sessions.

Following is an example of an NCST session interface definition:

```
*********************************************************************
*     NCST IP INTERFACES                                            *
*********************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),   *
               LINECB=CXSXLNK
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000
```

```
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSL0000,CXSS0*
               000),LOCADDR=1
*
```

Note: The NCST LU name (A04TOLU1 in this example) must match the LU name on the SNALINK LU0 DEVICE statement in *hlq*.PROFILE.TCPIP. See the example in “Step 2: Configure VTAM and SNALINK Applications” on page 1041 for more information.

Channel PU Interface Definition

Use this definition with channel PU interfaces (ESCON, BCCA, or CADS) to communicate with NCPROUTE using IP over CDLC devices with the MVS host.

Following is an example of channel PU interface definition using the ESCON channel type:

```
*********************************************************************
*   PHYSICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
*
A04PSOC1 GROUP LNCTL=CA,MONLINK=NO,NPACOLL=NO,XMONLNK=YES
               SPEED=144000000,SRT=(32768,32768)
*
A04S2240 LINE  ADDRESS=2240
A04P2240 PU    ANS=CONTINUE,PUTYPE=1
*
*********************************************************************
*    LOGICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
*
A04PSOCB GROUP LNCTL=CA,PHYSRSC=A04P2240,NPACOLL=NO,
               DELAY=0.2,MAXPU=16,MODETAB=AMODETAB,
               DLOGMOD=INTERACT,SPEED=144000000,
               SRT=(21000,20000),PUDR=YES,
               TIMEOUT=150.0,CASDL=0.0
*
A04LSOC2 LINE  ADDRESS=NONE,HOSTLINK=1
A04L2S1  PU    ADDR=01,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU1
A04L2S2  PU    ADDR=02,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU2
```

NCP Host Interface Definition

The IPOWNER statement in the NCP generation definition contains the TCP/IP host information and tells NCP which interface to use for NCPROUTE. The following are important keywords on this statement:

INTFACE
Specifies the name of the interface to the owning TCP/IP host that is running NCPROUTE.

HOSTADDR
Specifies the IP address of the owning TCP/IP. This address must match the IP address in the HOME statement in *hlq*.PROFILE.TCPIP data set for a SNALINK or IP over CDLC interface.

UDPPORT
Specifies the UDP port number for NCPROUTE. The default is 580. This port

number must match the NCPROUTE service port number defined in the *hlq*.ETC.SERVICES data set. See “Step 5: Update hlq.ETC.SERVICES” on page 1045 for more information.

The IPLOCAL statement in the NCP generation definition contains the NCP routing information for the local attached routes. During NCP generation, this information gets included in the Routing Information Table (RIT) which NCPROUTE uses to build the interface and routing tables. IPLOCAL routes are predefined as permanent or static to prevent modification by NCPROUTE. The following are important keywords on this statement:

INTFACE
Specifies the name of the locally attached interface.

LADDR
Specifies the IP address of the locally attached interface.

P2PDEST
For point-to-point interfaces only. Specifies the IP address of the remote end of the point-to-point link.

PROTOCOL
Specifies the type of protocol to be used for the interface. The default is RIP which indicates that the interface is RIP-managed by NCPROUTE.

SNETMASK
Specifies the subnetwork mask for a route to a network that is subnetted. Since RIP does not support variable subnetwork masking, this value must equal to the subnetwork mask of the route’s destination.

The IPROUTE statement in the NCP generation definition contains the NCP routing information for optional predefined routes. During NCP generation, this information gets included in RIT which NCPROUTE uses to add the routes to its routing tables. IPROUTE routes can be predefined as permanent or non-permanent for route management control by NCPROUTE. The following are important keywords on this statement:

INTFACE
Specifies the name of the locally attached interface for the route.

DESTADDR
Specifies the route’s destination IP address.

DISP
Specifies the disposition for the route. A disposition of PERM indicates that this route is a permanent route and will not be modified by NCPROUTE. The default is NONPERM.

HOSTRT
Indicates that this is a host route. The default is NO.

NEXTADDR
Specifies the IP address of the gateway through which the route can reach its destination. A value of zero indicates that there is no gateway.

The following example shows typical NCP RIP router generation source statements.

```
*********************************************************************
*       IP ROUTING DEFINITIONS                                      *
*********************************************************************
*
        IPOWNER INTFACE=NCSTALU1,HOSTADDR=9.67.116.66,              *
                NUMROUTE=(100,100,100),MAXHELLO=25,UDPPORT=580
```

```
*
        IPLOCAL LADDR=9.67.116.65,INTFACE=NCSTALU1,METRIC=1,          *
                P2PDEST=9.67.116.66,PROTOCOL=RIP,SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.88,INTFACE=TR88,METRIC=1,               *
                SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.92,INTFACE=TR92,METRIC=1,               *
                SNETMASK=FFFFF000
*
        IPROUTE DESTADDR=11.0.0.1,NEXTADDR=0,INTFACE=TR88,METRIC=2,   *
                DISP=PERM,HOSTRT=YES
        IPROUTE DESTADDR=12.0.0.0,NEXTADDR=13.0.0.1,INTFACE=TR92,     *
                METRIC=2,DISP=NONPERM
```

The following example shows IPOWNER and IPLOCAL statements for the ESCON channel PU interfaces in the configuration for NCP2 as shown in Figure 52 on page 1040.

```
************************************************************
*   IP ROUTING DEFINITIONS USING ESCON CHANNEL INTERFACES   *
************************************************************
*
IPOWNER INTFACE=SOCPU1,UDPPORT=580,NUMROUTE=(110,120,130),
        HOSTADDR=12.1.1.1
*
IPLOCAL LADDR=12.1.1.98,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
*
IPLOCAL LADDR=12.1.1.99,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.2,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
*
IPLOCAL LADDR=13.1.1.99,INTFACE=SOCPU2,METRIC=1,
        P2PDEST=13.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF80
```

Step 7: Configure the NCPROUTE Profile Data Set

To build the NCPROUTE profile, create a data set and specify its name in the //NCPRPROF DD statement in the NCPROUTE cataloged procedure. You can find a sample in *hlq*.SEZAINST(EZBNRPRF). Include configuration statements in this data set to define SNMP functions and to identify the NCPROUTE gateways data set. For more information, see “Chapter 22. Configuring Simple Network Management Protocol (SNMP)” on page 823.

RIP_SUPPLY_CONTROL *supply_control*

Specifies one of the following options on a server-wide basis:

- RIP1—Unicast/Broadcast RIP Version 1 packets (Default)
- RIP2B—Unicast/Broadcast RIP Version 2 packets (Not Recommended)
- RIP2M—Unicast/Multicast/Broadcast RIP packets (Migration)
- RIP2—Unicast/Multicast RIP Version 2 packets
- NONE—Disables sending RIP packets

Note: If RIP2 is specified, the RIP Version 2 packets are multicast over multicast-capable interfaces only. No RIP packets are sent over multicast-incapable interfaces. For RIP2M, the RIP Version 2 packets are multicast over multicast-capable interfaces and RIP Version 1 packets over multicast-incapable interfaces. For RIP2B, the RIP Version 2 packets are unicast or broadcast; this option is not recommended since host route misinterpretations by adjacent routers running RIP Version 1 can occur. For this reason, RIP2B may become obsolete in a future release. For point-to-point interfaces that are non-broadcast and multicast-incapable, the RIP Version 2 packets are unicast.

RIP_RECEIVE_CONTROL *receive_control*
Specifies one of the following options on a server-wide basis:

- RIP1—Receive RIP Version 1 packets only
- RIP2—Receive RIP Version 2 packets only
- ANY—Receive any RIP Version 1 and 2 packets (Default)
- NONE—Disables receiving RIP packets

Note: If the client NCP does not support variable subnetting, the default of ANY is changed to RIP1.

RIP2_AUTHENTICATION_KEY *authentication_key*
Specifies a plain text password *authentication_key* containing up to 16 characters. The key is used on a router-wide basis and can contain mixed case and blank characters. Single quotes (') can be included as delimiters to include leading and trailing blanks. The key will be used to authenticate RIP Version 2 packets and be included in the RIP updates for authentication by adjacent routers running RIP Version 2. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets. A blank key indicates that authentication is disabled. Following are examples of authentication passwords:

```
my password        (no leading or trailing blanks)
' my password '    (leading and trailing blanks)
''abc''            (single quotes part of password)
'     '            (5-character blanks)
```

SNMP_AGENT *host_name*
Specifies the host name or IP address of the host running an SNMP daemon. Only one NCPROUTE server can use a particular SNMP agent at a time.

SNMP_COMMUNITY *community_name*
Specifies a community name that SNMP applications must use to access data that the agent manages. Protect this information accordingly.

GATEWAY_PDS *dsname*
Specifies the optional partitioned data set that contains GATEWAY information for each client NCP. Quotation marks are not needed when specifying *dsname*. One member for each NCP client of this data set must be configured to match the NCP NEWNAME parameter with the **P** suffix which is the same as the NCP's RIT member name. See "Step 8: Configure the NCPROUTE Gateways Data Set (Optional)" on page 1051 for information on the defining the statements necessary for the members of this data set.

Note: You can use a semicolon in column 1 to permit comments in the profile. Blank lines are also permitted.

Figure 53 on page 1051 shows the relationship between the data set names specified in the NCPROUTE cataloged procedure and the NCPROUTE profile, as well as the relationship between the members of the gateways PDS and the ncpload PDS.

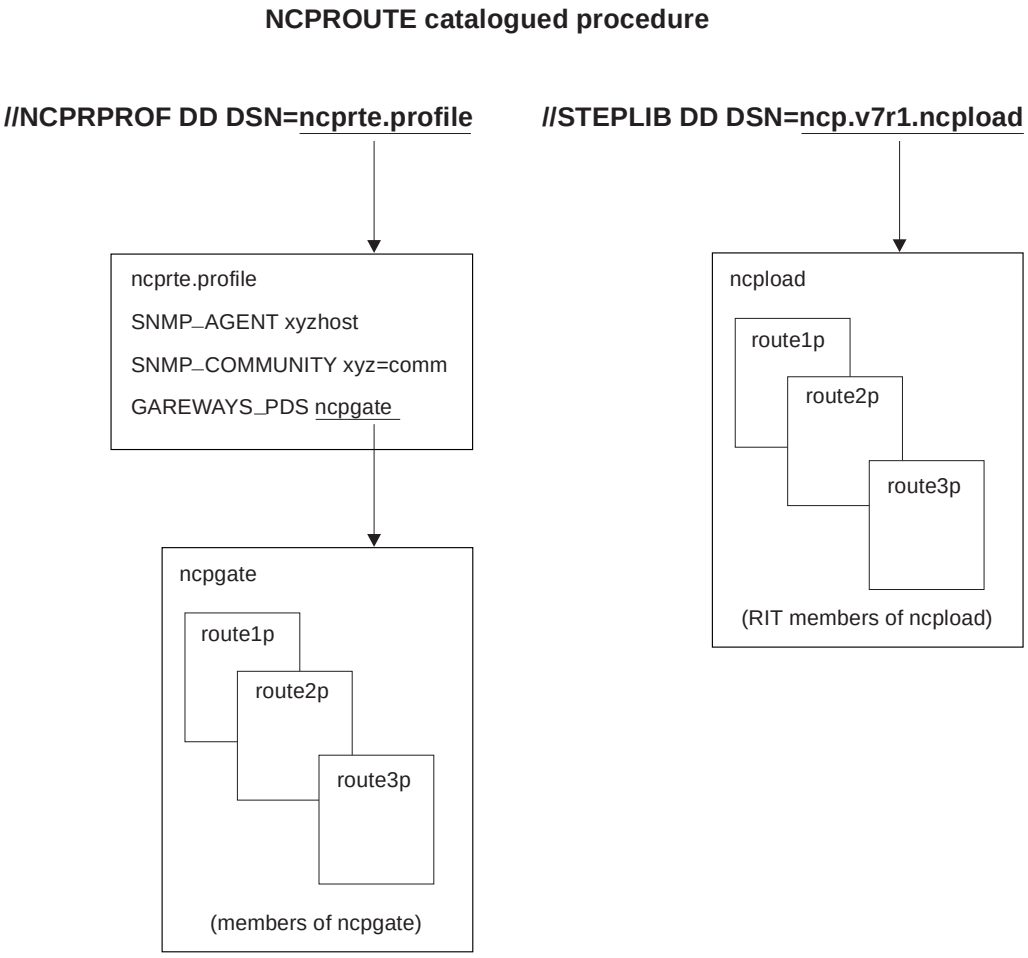


Figure 53. NCPROUTE Data Sets Relationship

Step 8: Configure the NCPROUTE Gateways Data Set (Optional)

The gateways data set is used to identify routes not defined in the NCP routing information table.

NCPROUTE and ROUTED require separate gateways data sets. The two servers cannot share the same data set. The NCPROUTE gateways data set is optional. However, if you use it, you must include the GATEWAY_PDS statement in the NCPROUTE profile to specify the gateway data set name. The NCPROUTE server queries the gateways data set for static routing information. It also dynamically receives routing information from the NCP client portion of this RIP router.

Allocate the gateways data set with partitioned organization (PO), a fixed block format (FB), a logical record length of 80 (LRECL), and any valid block size value for a fixed block, such as 3120.

A `passive` entry in the gateways data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the

routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

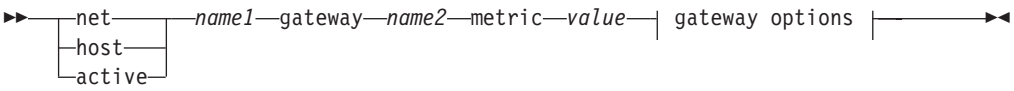
Note: The gateways data set is not related to the GATEWAY statement used in *hlq*.PROFILE.TCPIP data set.

To configure NCPROUTE, add an entry to the gateways data set for each route not defined in the NCP RIT. Use the options statement to define the characteristics of the routes in this member of the PDS.

Syntax Rules

- You can specify multiple GATEWAY statements.
- Keywords can be specified in mixed case.
- Blanks and comments are supported in the gateways data set. Comments are identified by a semicolon in column 1.
- GATEWAY statements must start in column 1.
- There should be no sequence numbers in the data set.

The syntax for the gateway statement is:



gateway options:

net
Indicates that the route goes to a network.

host
Indicates that the route goes to a specific host.

active
Indicates that the route to the gateway will be treated as a network interface.

name1
Can be either a symbolic name or the IP address of the destination network or host. *name1* must be specified as `active` if this is for an active gateway.

gateway
The parameters that follow this keyword identify the gateway or router for this destination.

name2
Can be either a symbolic name or the IP address of the gateway or router for this destination.

metric
The value that follows this keyword is the hop count to the destination.

value
Indicates the hop count to this destination. This number is an integer between 1 and 15, where 15 indicates that the network cannot be reached.

passive
A passive gateway does not exchange routing information. Information about the passive gateway is maintained in the local routing tables indefinitely and is only local to this NCPROUTE server. Passive gateway entries for indirect routes are not included in any routing information that is transmitted. Directly connected passive routes are included.

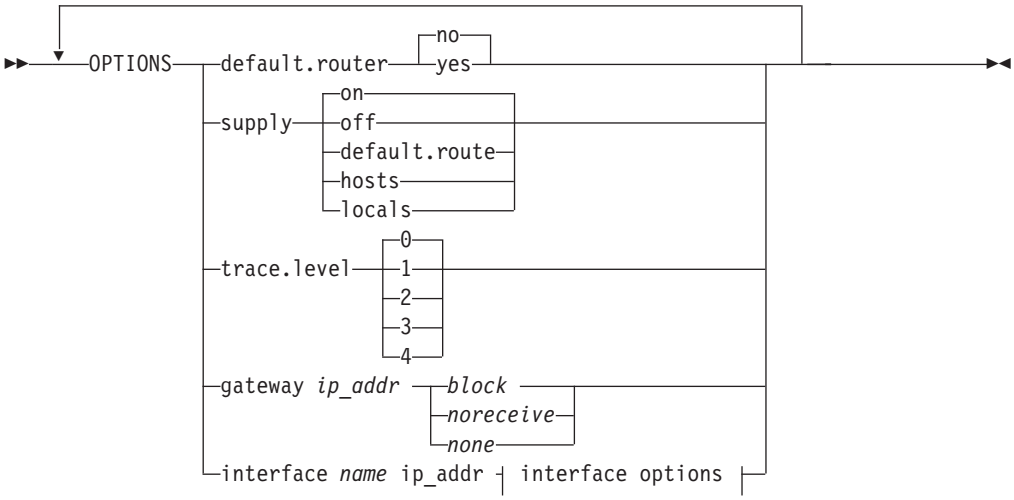
external
Indicates that entries for this destination should never be added to the routing table. The NCPROUTE server discards any routes for this destination that it receives from other routers. Only the destination field is significant, the gateway and metric fields are ignored.

active
Indicates that the router is treated as a network interface. An active gateway is a router that is running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

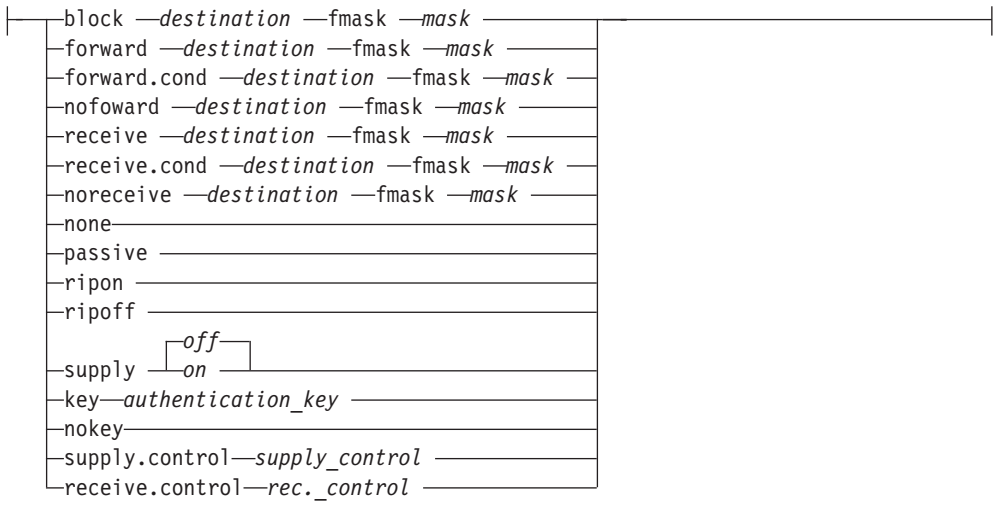
mask
A constant. The value that follows this key word is the subnet mask for the route.

subnetmask
A bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous in the network portion of the *subnetmask*. If the *subnetmask* is not specified, NCPROUTE will default the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, then the network class mask for the route is used.

The syntax for the options statement is:



interface options:



default.router

Enables the default router. When this option is specified, NCPROUTE adds a default route to its routing information and propagates it over all local interfaces. If the adjacent routers add the default route to their routing tables, NCPROUTE will receive all unknown packets from them and funnel them to a destination router, provided that a default route is defined. If this option is used, it is recommended that a default route to a destination router be defined on an IPROUTE statement in NCP generation definition or in the NCPROUTE gateways data set. See “Configuring a Default Route” on page 940.

yes This is the default router.

no This is not the default router.

interface

A constant. The parameters *name* and *ipaddr* follow this keyword.

name Specifies the name of the interface according to NCP clients NCP generation. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all interface names.

ipaddr Specifies the internet address of the interface associated with the interface name. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all internet addresses of the interfaces.

noreceive (or block)

If interface option, specifies that the *destination* route in the received RIP packets for this interface are to be ignored. If gateway option, specifies that routing table broadcasts from this gateway are to be ignored. This option is provided as a RIP input filter.

destination

Specifies that the destination route in network, subnetwork, or host format. A specification of an asterisk (*) indicates that all destination routes to be used with the *noforward* and *noreceive* options. This serves as a ″blackhole″ filter option which can be used to filter out all routes RIP packets over an interface and allow routes with specified *forward* and *receive* filter to be used.

fmask A constant. The value that follows this keyword is the filter mask for the route.

mask Optional bit mask (expressed in dotted-decimal form) defining the routing filter mask associated with the destination route. This mask is to

be used as an optional parameter to the forward and receive parameters to filter in/out multiple routes matching the mask of the destination route. This option can be used to define a single RIP input or output filter representing multiple routes as opposed to defining individual RIP input or output filters for each route.

forward
Specifies that the *destination* route in the RIP responses is to be forwarded to this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

forward.cond
Specifies that the *destination* route is to be forwarded to this interface only when the interface is active. In case of an interface outage, NCPROUTE will include the *destination* route in the RIP responses to other active interfaces. After recovery of an interface outage, NCPROUTE will resume to sending the destination route over this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

metric The metric associated with the cost of use for the link. When sending routing information over this link, NCPROUTE will use the *new_metric* value in the routing metrics for the routes that are advertised over this link. If this option is not used, the metric value that is used will be the value specified in the IPLOCAL statement of NCP generation definition. This option allows you to override the genned metric. If a metric of 1 is specified, a metric value of 1 will be used; this is the default cost for a directly-connected network. If a metric of 2 is specified, a metric value of 2 will be used. As the metric gets higher, the routes sent over this link become less preferred. The range is from 1 to 15. A metric of 1 is usually coded so that the routes sent over the interface will be the most preferred.

noforward
Specifies that the destination route in the RIP responses is not to be forwarded. This option is provided as a RIP output filter.

none If interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If asterisks (*) are specified for interface name and ipaddr, all options will be cleared from all interfaces. If gateway option, specifies that any RIP filter options for the gateway are to be turned off or reset. If an asterisk (*) is specified for the internet addresses, all gateway entries with gateway options will be cleared.

receive
Specifies that the destination route in the RIP responses is to be received over this interface only. This option is provided as a RIP input filter.

receive.cond
Specifies that the destination route is to be received over this interface only when is active. In case of an interface outage, NCPROUTE will allow the destination route in the RIP responses to be received over other active interfaces. This option is provided as a RIP input filter and can be used for inbound and outbound traffic splitting.

ripoff (or passive)
Specifies that RIP is disabled for this interface. NCPROUTE will not supply nor receive RIP updates.

ripon Specifies that RIP is enabled for this interface. RIP responses will be allowed to be sent or received over this interface.

supply
Defines the supply routing setting. The default is on. This option is provided as a RIP input and output filter.

on Supply routing information for this NCP client or interface.

off Suppresses supply of routing information for this NCP client or interface. NCPROUTE will continue to receive routing updates.

default.route
Supply the default route only for this NCP client. When this option is specified, *yes* is internally set for the *default.router* option. This option is provided as a RIP output filter.

hosts Supply routing information with host routes added.

locals Supply only local (directly connected) routes.

trace.level
Specifies the trace level to be used for this NCP client. The default is 0.

0 Do not allow tracing.

1 Activates tracing of actions by the NCPROUTE server.

2 Activates tracing of actions and packets sent or received.

3 Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced and are displayed whenever an interface goes inactive.

4 Activates tracing of actions, packets sent or received, packet history, and packet contents. The packet contents displays the RIP network routing information.

key
Specifies a plain text password authentication key containing up to 16 characters to be used for this interface and is used to override the server-wide setting defined in the NCPROUTE profile. It can contain mixed case and blank characters. Single quotes (') can be included as delimiters to include leading and trailing blanks. A null or blank key indicates that the server-wide key will be used as the default. For examples on authentication passwords, see the RIP2_AUTHENTICATION_KEY statement in "Step 7: Configure the NCPROUTE Profile Data Set" on page 1049.

nokey
Specifies that authentication is disabled for this interface even though the server-wide specification from the RouteD profile is defined.

supply.control
Specifies that the RIP *supply_control* is to be used for this client or interface and is used to override the NCPROUTE profile setting.

supply_control
Specifies one of the RIP supply control options. The default is set to the NCPROUTE profile setting. Valid options are RIP1, RIP2B, RIP2M, RIP2, and NONE. For more information on this option, see "Step 7: Configure the NCPROUTE Profile Data Set".

receive.control
Specifies that the RIP *receive_control* is to be used for this client or interface and is used to override the NCPROUTE profile setting.

receive_control
Specifies one of the RIP receive control options. The default is set to the NCPROUTE profile setting. Valid options are RIP1, RIP2, ANY, and NONE. For more information on this option, see ″Step 7: Configure the NCPROUTE Profile Data Set″.

gateway
A constant. The value that follows this keyword identifies the gateway or router.

ipaddr
If interface option, specifies the internet address of the interface associated with the interface name. If gateway option, specifies the gateway address of the adjacent router. A specification of an '*' applies to all gateway addresses.

none
If interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If gateway option, specifies that any RIP filter options for this gateway are to be turned off or reset. A specification of an asterisk (*) indicates all interface internet addresses or all gateway addresses.

noreceive (or block)
If interface option, specifies that the *destination* route in the RIP responses propagates is not to be received over this interface only. If gateway option, specifies that no RIP packets are to be received from the specified gateway address of the adjacent router. This option provides as a RIP input filter.

Note: All traces will go to a standard output referred to in the //SYSPRINT DD statement in the NCPROUTE cataloged procedure.

The options can be specified in any order. For example:

```
options default.router yes supply on trace.level 2
options interface ETH1 10.1.1.1 passive
options interface ETH1 10.1.1.1 supply off
options interface TR1 9.67.112.25 metric 2
options interface TR1 9.67.112.25 forward 11.0.0.0
options interface TR1 9.67.112.25 forward.cond 12.0.0.0
options interface TR1 9.67.112.25 block 9.1.0.0
options interface TR1 9.67.112.25 supply.control rip1
options interface ETH1 10.1.1.1 receive.control rip2
options interface ETH2 9.1.1.1 forward 9.2.0.0 fmask 255.255.0.0
options interface ETH1 10.1.1.1 none
options interface * * none
options gateway 9.67.112.77 noreceive
options gateway 9.67.112.77 none
options gateway * none
```

Table 46 shows how the above parameters affect the advertising algorithm for routes in RIP responses to adjacent routers. The parameters can be used as router-wide RIP output filters. To configure interface-wide RIP input and output filters, see the OPTIONS statement in the NCPROUTE Gateways configuration data set.

Table 46. NCPRoute Options

Option	Host Routes	Network Routes	Advertise as Default Router	Local Routes	Unreachable Routes
default.router yes		Yes	Yes	Yes	Yes

Table 46. NCPRoute Options (continued)

Option	Host Routes	Network Routes	Advertise as Default Router	Local Routes	Unreachable Routes
Supply default.route			Yes		Yes
Supply hosts	Yes	Yes		Yes	Yes
Supply locals				Yes	Yes
Supply on		Yes		Yes	Yes
Supply off					
None		Yes		Yes	Yes

Configuring a Passive Route

Figure 54 illustrates an NCPROUTE configuration using NCP as the destination hosts. In other configurations, destination hosts might not necessarily be NCPs.

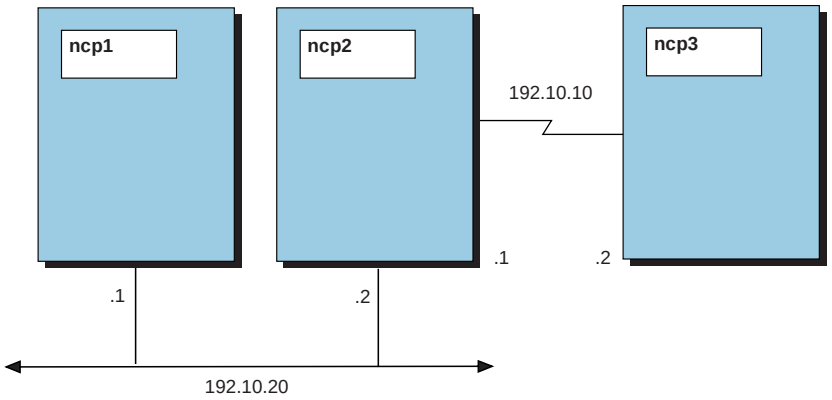


Figure 54. NCPROUTE Configuration Example

Using Figure 54, assume that your NCP client `ncp1` is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients, `ncp2` and `ncp3`, are not running a RIP server. Also assume that permanent routes to `ncp2` and `ncp3` are not defined with the IPROUTE definitions in the NCP generation definition for `ncp1`. Your NCPROUTE server can not learn a route to `ncp3`, because `ncp2` is not running a RIP server. Your NCPROUTE server sends routing updates to `ncp3` over the link to `ncp2`, but never receives a routing update from `ncp2`. After 180 seconds, your NCPROUTE server deletes the route to `ncp2`. This problem is inherent to the RIP protocol and cannot be prevented. Therefore, you need to add a `passive` route to `ncp3` in the NCPROUTE gateways data set for the NCP client `ncp1`. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile.

You can use either of the following gateway statements:

```
host  ncp3         gateway  ncp2         metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, because `ncp2` is not running an RIP server supported by NCPROUTE, you need to add a directly-connected `passive` route as follows:

```
host  ncp2         gateway  ncp1         metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the NCP generation.

Assume that your NCP client is now ncp2 and is running a NCPROUTE server. ncp1 is also running a RIP server, but ncp3 is not. Your NCPROUTE server sends routing information updates to ncp3 over the link to ncp3 but never recieves a routing update from ncp3. After 180 seconds, your NCPROUTE server deletes the route to ncp3.

You should add a passive route to ncp3 as follows:

```
   host  ncp3        gateway  ncp2        metric 1  passive
```

ncp1 cannot reach ncp3 unless a passive routing entry is added to ncp1. For example:

```
host   ncp3    gateway  ncp2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an External Route

Using Figure 54, assume that your NCP client ncp1 is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients ncp2 and ncp3, are also running a RIP server. Your NCPROUTE server normally learns a route to ncp3 from ncp2, because ncp2 is running a RIP server. You might not want ncp1 to route to ncp3 for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your NCPROUTE server from adding a route to ncp3, add an external route to the NCPROUTE gateways data set. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile. You can use either of the following gateway statements:

```
  host  ncp3         gateway   ncp2        metric 2  external

  host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an Active Gateway

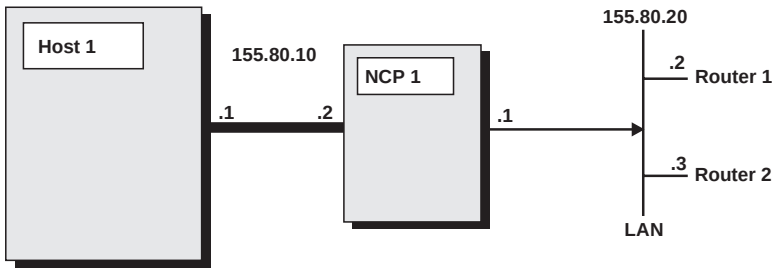


Figure 55. Configuring an Active Gateway

As shown in Figure 55, assume that your NCP client is ncp1, which is channel-attached to an MVS host running an NCPROUTE server and that it has a network attachment adapter that does not support link-level broadcasting or one

that does not support ARP processing. Also, assume that there are routers router1 and router2 on the local area network. Since the IP addresses router1 and router2 are unknown by ncp1, they have to be manually configured in NCPROUTE for NCPROUTE to communicate with them. Configuring active gateways for router1and router2 as remote network interfaces enables NCPROUTE to send RIP responses to the target addresses.

1. Specify IP addresses for each network adapter (without link-level broadcasting or ARP support) attached to the local network in the NCP client according to the NCP generation definition. For example, 155.80.20.1 is the IP address for the local network adapter attachment in ncp1.
2. Define active gateways for the remote routers in the NCPROUTE gateways data set specified on the GATEWAYS_PDS statement in the NCPROUTE profile:

```
active active gateway 155.80.20.2 metric 1 active
active active gateway 155.80.20.3 metric 1 active
```

From these active gateway addresses, NCPROUTE will use them as the destination addresses to send RIP responses to the remote routers. In addition, NCPROUTE will continue to receive RIP responses from the active gateways over the NCP client.

Configuring a Default Route

A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the IPROUTE statement in the NCP generation definition. For example, if the default destination router has a gateway address 9.67.112.1, an IPROUTE statement might look like:

```
IPROUTE  DESTADDR=0.0.0.0,NEXTADDR=9.67.112.1,INTFACE=TR88,
         METRIC=1,DISP=PERM
```

An easier way would be to use the passive route definition specified in the NCPROUTE gateways data set for the NCP client. For example, the gateways entry would look like:

```
net 0.0.0.0 gateway 9.67.112.1 metric 1 passive
```

Only one default route to a destination gateway or router can be specified. For a NCP client, NCPROUTE currently does not support multiple default routes.

Configuration Examples

The following example shows the contents of an NCPROUTE gateways data set containing multiple entries:

```
options default.router no trace.level 4 supply on
net testnet gateway 9.0.0.100 metric 1 passive
net 2.0.0.2 gateway 9.0.0.101 metric 2 external
host 2.0.0.3 gateway 9.0.0.102 metric 3 passive
host 2.0.0.4 gateway 9.0.0.103 metric 2 external
active active gateway 2.0.1.1 metric 1 active
```

In the second entry, the route indicates that NCPROUTE can reach network testnet through the gateway 9.0.0.100, and that it is 1 hop away. This passive route is not broadcast to other RIP routers.

In the third entry, the route indicates that NCPROUTE can reach network 2.0.0.2 through the gateway 9.0.0.101, and that it is 2 hops away. Because this route is external, NCPROUTE should not add routes for this network to the routing tables and routes received from other RIP routers for this network should not be accepted.

In the fourth entry, the route indicates that NCPROUTE can reach host 2.0.0.3 through gateway 9.0.0.102, and that it is 1 hop away. This passive route is not broadcast to other RIP routers.

In the fifth entry, the route indicates that NCPROUTE can reach host 2.0.0.4 through gateway 9.0.0.103, and that it is 2 hops away. Because this route is external, NCPROUTE should not add routes for this network to the routing tables, and routes received from other RIP routers for this network should not be accepted.

The sixth entry shows an active gateway. Note that it is specified as the last entry in the data set.

Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways data set if the default route is not defined in a NCP client's generation definition.

Step 9: Define a Directly Connected Host Route for the NCST Session

If you are not using RouteD, you need to configure a directly-connected static host route using the GATEWAY statement in *hlq*.PROFILE.TCPIP. For example, if you are using SNALINK as the host route and have the IP addresses shown in Figure 52 on page 1040, the GATEWAY statement might look like this:

```
GATEWAY
; net_number first_hop link_name packet_size     subnet_mask  subnet_value
  9.67.116.65     =    SNALINK     32758           HOST
```

See "GATEWAY Statement" on page 193 and the GATEWAY syntax information in "Step 8: Configure the NCPROUTE Gateways Data Set (Optional)" on page 1051 for more information about configuring GATEWAYS statements.

Note: The host routes on the MVS host are managed by TCP/IP as defined on the GATEWAY statement or by RouteD as defined on the BSDROUTINGPARMS statement. NCPROUTE does not manage the host routes on the MVS host. It only manages the routes on the NCP clients.

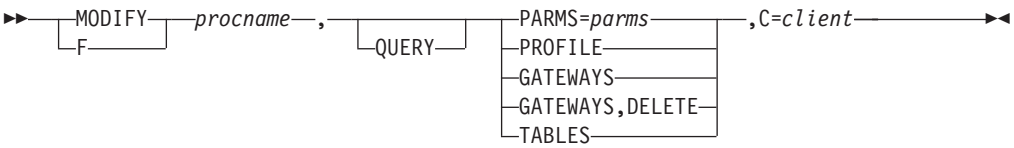
Controlling the NCPROUTE Address Space with the MODIFY Command

You can control most of the functions of the NCPROUTE address space from the operator's console using the MODIFY command. Following is the correct syntax and valid parameters for the NCPROUTE address space.

MODIFY Command—NCPROUTE Address Space

Use the MODIFY command to pass parameters to the NCPROUTE address space.

Syntax



Parameters

procname
The member name of the cataloged procedure used to start the NCPROUTE server.

client
The target client NCP's name or IP address. A value of zero indicates all clients. The default will be the first client that has an established session with NCPROUTE. This parameter can be issued at once to indicate that NCPROUTE is to process modify commands for this client or for all clients. If C=0 is specified or if NCPROUTE does not have any active sessions with its client(s), then only the parameters PARMS= and PROFILE are allowed to be processed.

parms
Any one or more of the following separated by a space. Enclosing the *parms* specified in single quotes or preceeding by a slash (/) is optional.

-g Enable default router.

-gq Disable default router.

-f Flush all indirect routes known from IP routing tables.

-fh Flush all indirect host routes known from IP routing tables.

-h Include host routes in addition to network-specific router for the RIP responses.

-hq Disable supply host routes.

-s Enable supply routing information.

-sd Enable supply default route only.

-sdq Disable supply default route only.

-sl Enable supplying of only local (directly connected) routes.

-slq Disable supplying of only local (directly connected) routes.

-sq or -q
Disable supply routing information.

-t Enable or disable traces. Up to 4 -t parms are allowed.

-tq Disable all traces.

-dp Enable debug packets trace.

-dq Disable all debug traces.

PROFILE
Reread the NCPROUTE PROFILE data set.

QUERY
Queries the current target client NCP's name or IP address.

GATEWAYS(,DELETE)
Reread the NCP's client's GATEWAYS data set member. If *DELETE* is specified, all routes listed in the data set are deleted.

TABLES
Displays NCPROUTE's internal IP routing and interface tables for diagnosis.

Consider the following when coding the parms:

Notes:

1. Enclosing quotes for the parms are optional.
2. Enclosing / for the parms is optional for example, parms=/-t -t).
3. If the c= parameter cannot be specified in one command, issue the modify command with this parameter alone, following another modify command for other parameters.
4. For -f or -fh parameters, only the indirect routes known by NCPROUTE are flushed:

Table 47 shows how the above parameters affect the advertising algorithm for routes in RIP responses to adjacent routers.

Note: The modify parameters correspond to the parameters in the OPTIONS statement of NCPROUTE Gateways data set.

Table 47. NCPROUTE Modify Parameters

Parameter	NCPROUTE GATEWAY Option	Host Routes	Network Routes	Advertise as Default Router	Local Routes	Unreachable Routes
-g	default router yes		Yes	Yes	Yes	Yes
-h	Supply local hosts	Yes	Yes		Yes	Yes
-s	Supply on		Yes		Yes	Yes
-sd	Supply default route			Yes		Yes
-sl	supply locals				Yes	Yes
-sq or -q	supply off					
None	None		Yes		Yes	Yes

Examples

```
F NCPROUT,GATEWAYS,c=NCP4
F NCPROUT,PARMS=-t -t -t -t,c=NCP1
F NCPROUT,PARMS=-tq, c=9.67.116.65
F NCPROUT,PARMS=c=10.1.1.99
F NCPROUT,PROFILE
F NCPROUT,PARMS=-tq
```

```
F NCPROUT,GATEWAYS,DELETE
F NCPROUT,PARMS,c=0
F NCPROUT,PARMS='/ -s -g'
F NCPROUT,PARMS=-h,PROFILE,GATEWAYS
```

Chapter 29. Configuring the OS/390 Unix PORTMAP Address Space

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the OS/390 UNIX PORTMAP address space, which runs the Portmapper function.

Configuration Process

Steps to configure OS/390 UNIX PORTMAP:

1. Specify PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.

Step 1: Specify the PORT Statements in PROFILE.TCPIP

To ensure that port UDP 111 and TCP 111 is reserved for the OS/390 UNIX PORTMAP server, add the name of the member containing the PORTMAP catalogued procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  111 UDP OMVS     ; Portmapper Server
  111 TCP OMVS     ; Portmapper Server
```

See “PORT Statement” on page 229 for more information about the PORT statement.

Step 2: Update the PORTMAP Cataloged Procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(OPORTPRC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Change the data set names as required.

PORTMAP Cataloged Procedure (OPORTPRC)

```
//PORTMAP PROC
//*
//*  OpenEdition MVS Portmapper Server main process
//*  Resulting address space name will be PORTMAP1, when
//*  we use this method to start the portmapper
//*
//*  OPORTMAP is in SYS1.TCPIP.SEZALINK (on the LINKLST)
//*
//PORTMAP EXEC PGM=OPORTMAP,REGION=40M,
//         PARM='POSIX(ON),ALL31(ON)/'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//         PEND
```

Starting the PORTMAP Address Space

There are two ways to start the portmapper as an OS/390 UNIX socket application:

- From the OS/390 shell
- As a started task.

To start the portmapper from the OS/390 shell, the user ID must be an authorized superuser. The authorized superuser ID can issue “oportmap &” to start the portmapper. For the authorization procedure, see OS/390 UNIX System Services Planning.

You can also start PORTMAP as a started task with the START command as follows:

```
START PORTMAP
```

Note: If your system is using the Network File System (NFS) server, see *Customizing and Operating the Network File System Server* for more information.

Chapter 30. Configuring the PORTMAP Address Space

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the PORTMAP address spaces, which runs the Portmapper function.

Portmapper is the software that supplies client programs with the port numbers of server programs. Clients contact server programs by sending messages to port numbers where receiving processes receive the message. Because you make requests to the port number of a server rather than directly to a server program, client programs need a way to find the port number of the server programs they wish to call. Portmapper standardizes the way clients locate the port number of the server programs supported on a network.

Configuration Process

Steps to configure PORTMAP:

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.
3. Define the data set for well-known procedure names.

Step 1: Specify the AUTOLOG and PORT Statements in PROFILE.TCPIP

If you want the PORTMAP server to start automatically when the TCPIP address space is started, you should include PORTMAP in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  PORTMAP
ENDAUTOLOG
```

Note: If your system is using the Network File System (NFS) server, you ***must*** start the PORTMAP address space. See *Customizing and Operating the Network File System Server* for more information.

To ensure that port UDP 111 and TCP 111 is reserved for the PORTMAP server, also add the name of the member containing the PORTMAP catalogued procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  111 UDP PORTMAP
  111 TCP PORTMAP
```

See “AUTOLOG Statement” on page 137 for more information about the AUTOLOG statement. See “PORT Statement” on page 229 for more information about the PORT statement.

Step 2: Update the PORTMAP Cataloged Procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(PORTPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify PORTMAP parameters, and change the data set names, as required.

PORTMAP Cataloged Procedure (PORTPROC)

```
//PORTMAP  PROC MODULE=PORTMAP,PARMS=''
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB029
//*
//*        5655-HAL (C) Copyright IBM Corp. 1989, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//PMAP     EXEC PGM=&MODULE,
//           PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR

//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 3: Define the Data Set for Well-Known Procedure Names

TCP/IP for MVS includes a data set that contains a list of commonly used procedure names. This data set is used by the RPCINFO command to resolve remote program numbers into their equivalent program names.

To create the data set, copy the ETCRPC member of *hlq*.SEZAINST to the default data set called *hlq*.ETC.RPC. If a user has a *user_id*.ETC.RPC data set defined, it takes precedence over the preceding data set.

Normally, you would not change this data set except to add a new application to the list. To add a new application, add a line that contains the following items:

- The *program_name* of the new application or procedure
- The *program_number* of the new application or procedure
- Any comments regarding the description of the program

The items are variable in format, each separated by a blank.

The *hlq*.SEZAINST(ETCRPC) data set contains the well-known procedure names. Following is the ETCRPC sample.

```
#
#   rpc   1.2   86/10/07
#
#   COPYRIGHT = NONE.
#
portmapper    100000    portmap sunrpc
rstatd        100001    rstat rup perfmeter
rusersd       100002    rusers
nfs           100003    nfsprog
ypserv        100004    ypprog
mountd        100005    mount showmount
ypbind        100007
walld         100008    rwall shutdown
yppasswdd     100009    yppasswd
etherstatd    100010    etherstat
rquotad       100011    rquotaprog quota rquota
sprayd        100012    spray
3270_mapper   100013
rje_mapper    100014
selection_svc 100015    selnsvc
database_svc  100016
rexd          100017    rex
alis          100018
sched         100019
llockmgr      100020
nlockmgr      100021
x25.inr       100022
statmon       100023
status        100024
#
# NDB Program numbers added for more useful rpcinfo output
# Values are in decimal. Corresponding hexadecimal values
# for the NDB port manager is X'20000020' and for the NDB
# servers are from X'20000021' to X'20000084'.
#
ndbportmgr    536870944
ndbserver1    536870945
ndbserver2    536870946
ndbserver3    536870947
ndbserver4    536870948
ndbserver5    536870949
ndbserver6    536870950
ndbserver7    536870951
ndbserver8    536870952
ndbserver9    536870953
ndbserver10   536870954
ndbserver11   536870955
ndbserver12   536870956
ndbserver13   536870957
ndbserver14   536870958
ndbserver15   536870959
ndbserver16   536870960
ndbserver17   536870961
ndbserver18   536870962
ndbserver19   536870963
ndbserver20   536870964
ndbserver21   536870965
ndbserver22   536870966
ndbserver23   536870967
ndbserver24   536870968
ndbserver25   536870969
ndbserver26   536870970
ndbserver27   536870971
ndbserver28   536870972
```

```
ndbserver29   536870973
ndbserver30   536870974
ndbserver31   536870975
ndbserver32   536870976
ndbserver33   536870977
ndbserver34   536870978
ndbserver35   536870979
ndbserver36   536870980
ndbserver37   536870981
ndbserver38   536870982
ndbserver39   536870983
ndbserver40   536870984
ndbserver41   536870985
ndbserver42   536870986
ndbserver43   536870987
ndbserver44   536870988
ndbserver45   536870989
ndbserver46   536870990
ndbserver47   536870991
ndbserver48   536870992
ndbserver49   536870993
ndbserver50   536870994
ndbserver51   536870995
ndbserver52   536870996
ndbserver53   536870997
ndbserver54   536870998
ndbserver55   536870999
ndbserver56   536871000
ndbserver57   536871001
ndbserver58   536871002
ndbserver59   536871003
ndbserver60   536871004
ndbserver61   536871005
ndbserver62   536871006
ndbserver63   536871007
ndbserver64   536871008
ndbserver65   536871009
ndbserver66   536871010
ndbserver67   536871011
ndbserver68   536871012
ndbserver69   536871013
ndbserver70   536871014
ndbserver71   536871015
ndbserver72   536871016
ndbserver73   536871017
ndbserver74   536871018
ndbserver75   536871019
ndbserver76   536871020
ndbserver77   536871021
ndbserver78   536871022
ndbserver79   536871023
ndbserver80   536871024
ndbserver81   536871025
ndbserver82   536871026
ndbserver83   536871027
ndbserver84   536871028
ndbserver85   536871029
ndbserver86   536871030
ndbserver87   536871031
ndbserver88   536871032
ndbserver89   536871033
ndbserver90   536871034
ndbserver91   536871035
ndbserver92   536871036
ndbserver93   536871037
ndbserver94   536871038
ndbserver95   536871039
```

```
ndbserver96   536871040
ndbserver97   536871041
ndbserver98   536871042
ndbserver99   536871043
ndbserver100  536871044
```

Starting the PORTMAP Address Space

If you did not include PORTMAP in the AUTOLOG statement, you can start PORTMAP with the START command. For example:

```
START PORTMAP
```

PORTMAP must be started before NDB can run.

Chapter 31. Configuring the NCS Interface

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the Network Computing System (NCS).

Understanding the NCS Interface

NCS is the Remote Procedure Call (RPC) implementation of Apollo’s Network Computing Architecture (NCA**).

NCS includes:
1. RPC runtime library
2. Location broker
3. Network Interface Definition Language (NIDL) compiler

The RPC runtime library and the location broker provide runtime support. Together these two elements make up the Network Computing Kernel (NCK) which includes all the software necessary to run a distributed application. The NIDL compiler is a tool for developing NCS-based applications.

In order to execute NCS applications in an MVS environment, you must start a local location broker (LLBD). One of the hosts in your TCP/IP network must also start the global location broker (GLBD). Both the LLBD and the NRGLBD maintain information about active NCS server applications.

Understanding the LLBD Server

The LLBD manages the LLB database which stores information for the NCS-based servers running on this host.

Your host must run LLBD if you want to support the location broker forwarding function or allow remote access to the LLB database. The LLBD function must be started on the host before any other NCS-based servers are started and before the NRGLBD is started.

The LLB database is stored in the data set ADM@SRV.LLB@LL.DATABASE. It is not created until an entry is registered with the LLBD.

Understanding the NRGLBD Server

The NRGLBD manages the NCS global location broker (GLB) database. The GLB helps clients locate servers on a network or internet.

There are two versions of the GLB daemon: replicated GLBD and NRGLBD. The replicated GLBD is only available on Apollo, SunOS, and Ultrix systems. For other systems, such as IBM, only the NRGLBD is available. The advantage of replicated GLBD over NRGLBD is that the GLBD can be run at the same time on several network hosts, providing greater availability in the event that one of the hosts running GLBD fails or if there is a partial network failure.

You cannot use the NRGLBD on your system if:
- The replicated version of GLBD can run on some other host in your network
- Another host in your network is already running NRGLBD

The GLB database is stored in a data set ADM@SRV.LLB@LG.DATABASE. This data set is not created until an entry is registered with the NRGLBD.

The record structure for the LLBD and the NRGLBD databases is identical.

Configuration Process

This chapter describes how to configure the NCS server.

Steps to configure NCS:
1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP
2. Update the NRGLBD cataloged procedure in *hlq*.SEZAINST(NRGLBD)
3. Update the LLBD cataloged procedure in *hlq*.SEZAINST(LLBD)

For more information about NCS, see *Network Computing System Reference Manual*

Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP

If you want LLBD and NRGLBD to start automatically when the TCPIP address space is started, then you should include the names of the members containing the LLBD and NRGLBD catalogued procedures in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

The LLBD must be running before you start NRGLBD. Therefore, you must put LLBD before NRGLBD in the AUTOLOG statement.

```
AUTOLOG
  LLBD
  NRGLBD
ENDAUTOLOG
```

To ensure that port UDP 135 is reserved for the LLBD server, add the name of the member containing the LLBD catalogued procedure to the PORT statement in the *hlq*.PROFILE.TCPIP data set.

```
PORT
  135 UDP LLBD
```

Note: OS/390 UNIX DCE also uses port 135 and therefore cannot be used concurrently with NCS.

See “AUTOLOG Statement” on page 137 for more information about the AUTOLOG statement. See “PORT Statement” on page 229 for more information about the PORT statement.

Step 2: Update the NRGLBD Cataloged Procedure

Update the NRGLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(NRGLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions.

NRGLBD Cataloged Procedure (NRGLBD)

Following is the sample NRGLBD cataloged procedure:

```
//NRGLBD   PROC MODULE=NRGLBD,PARMS=''
//*
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB02D
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//*
//NRGLBD   EXEC PGM=&MODULE,
//         PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized. Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR

//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 3: Update the LLBD Cataloged Procedure

Update the LLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(LLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions.

LLBD Cataloged Procedure (LLBD)

Following is the sample LLBD cataloged procedure:

```
//LLBD     PROC MODULE=LLBD,PARMS=''
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB02E
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
```

```
//*        See IBM Copyright Instructions
//*
//LLBD     EXEC PGM=&MODULE,
//         PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.  Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR

//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
```

Chapter 32. Configuring the Network Database (NDB) System

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the Network DataBase (NDB) System. NDB provides access to mainframe relational database systems using TCP/IP and the remote procedure call (RPC) protocol. This allows interoperability among a variety of workstation and mainframe users with DB2 and MVS. End users in a VM, MVS, DOS, OS/2, RISC System/6000® AIX, or Sun UNIX Microsystems environment can issue SQL statements interactively or invoke NDB services from within a C application program. NDB services can then be used to pass SQL statements to DB2 and to handle responses from DB2.

The Network DataBase (NDB) System consists of:

- The Network Database (NDB) Port Manager
- The NDB Port Client
- 1–100 NDB Servers
- NDB Clients

Note: The NDB System requires the Portmapper to run. See “Chapter 30. Configuring the PORTMAP Address Space” on page 1067 for further information.

Configuration Process

This section describes how to configure the NDB System. You can find instructions for starting NDB in “Starting NDB” on page 1089.

Steps to Configure the NDB System

1. Update the NDB setup sample job (NDBSETUP).
2. Run the NDB setup sample job.
3. Update and install the DB2 sample connection exit routine (DSN3SATH), if necessary.
4. Update the PORTS cataloged procedure (PORTSPRC).
5. Update the PORTC cataloged procedure (PORTCPRC).
6. Create the NDB clients.

Note: Repeat steps 1 and 2, and step 3 if necessary, to configure each DB2 subsystem that is to be accessed by NDB servers.

Step 1: Update the NDB Setup Sample Job

Update the NDBSETUP sample job by copying the sample in *hlq*.SEZAINST(NDBSETUP) to a partitioned data set (PDS) or sequential data set and modifying it to suit your local installation. Change the DD statements as required.

To restrict the use of NDB to selected users, modify the GRANT statement in NDBSETUP job replacing PUBLIC with a list of the specified TSO user IDs to whom you want to grant access. See the *IBM DATABASE2 SQL Reference* for correct statement syntax.

THE NDBSETUP job binds the DBRM DBUTIL2 into the DB2 subsystem specified. In previous releases, the DBRM was bound using the plan name DBUTIL2. For V3R2, the DBRM is bound using the plan name EZAND320. By using different plan names for each release, you can have different levels of TCP/IP similutaneously accessing the same DB2 subsystems.

Note: NDBSETUP binds the DBRM DBUTIL2 with the isolation level of cursor stability: ISOLATION(CS) on the BIND command. If a higher degree of unit of work integrity is needed than is provided by an isolation level of cursor stability, this can be changed to an isolation level of repeatable read: ISOLATION(RR) on the BIND command. Refer to the *IBM DATABASE2 Application Programming and SQL Guide* for information regarding isolation level settings and their effects. Check with your database administrator regarding isolation level policies at your site.

NDB Set Up Cataloged Procedure (NDBSETUP)

The NDBSETUP sample contains statements that correspond with the other NDB samples in this chapter.

```
//NDBSETUP JOB NDBSETUP,
//            CLASS=A,
//            NOTIFY=&SYSUID
//*
//*  TCP/IP for MVS
//*  SMP/E Distribution Name: EZAEB032
//*
//*  5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*  All rights reserved.
//*  US government users restricted rights -
//*  use, duplication or disclosure restricted
//*  by GSA ADP schedule contract with IBM Corp.
//*  See IBM copyright instructions
//*
//*  This JCL causes the bind of the DBUTIL2 DBRM to the
//*  specified DB2 subsystem and allows execution of the
//*  DBUTIL2 plan by PUBLIC.
//*
//*  This plan is used by the Network Database Server.
//*
//*  Usage notes:
//*
//*    1.  This job must be executed from a user ID that has
//*        the authority to bind the DBUTIL2 plan.
//*
//*    2.  Change the STEPLIB DD statement in the NDBIND and
//*        NDBGRANT steps to reflect the DB2 DSNLOAD data set
//*        installed on your system.
//*
//*    3.  Change the DB2 subsystem name in the NDBIND and
//*        NDBGRANT steps from "SYSTEM(xxx)" to the
//*        installation-defined DB2 subsystem name.
//*
//*    4.  Change the high level qualifier of SEZADBRM in
//*        the library parameter in the NDBIND step to
//*        match your system installation.
//*
//*    5.  Change the PLAN parameter defined in the RUN statement
```

```
//*        in the NDBGRANT step to reflect the DB2 release level
//*        installed on your system (e.g. DSNTIA31).
//*
//*    6.  Change the LIBRARY parameter defined in the RUN statement
//*        in the NDBGRANT step from "xxxxxx.RUNLIB.LOAD" to reflect
//*        the library where the DSNTIAD program resides.
//*
//NDBIND   EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN='xxxxxx.DSNLOAD',DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 BIND ACQUIRE(USE) -
      ACTION(REPLACE) -
      CACHESIZE(1024) -
      CURRENTDATA(NO) -
      EXPLAIN(NO) -
      ISOLATION(CS) -
      LIBRARY('TCPIP.SEZADBRM') -
      MEMBER(DBUTIL2) -
      NODEFER(PREPARE) -
      PLAN(DBUTIL2) -
      QUALIFIER(SYSADM) -
      RELEASE(COMMIT) -
      VALIDATE(RUN) -
      RETAIN
 END
//*
//NDBGRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN='xxxxxx.DSNLOAD',DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN  SYSTEM(xxx)
 RUN  PROGRAM(DSNTIAD) -
      PLAN(DSNTIAxx) -
      LIBRARY('xxxxxx.RUNLIB.LOAD')
 END
//SYSIN    DD  *
 GRANT EXECUTE ON PLAN DBUTIL2 TO PUBLIC;
//*
```

Step 2: Run the NDB Setup Job

Run NDBSETUP to bind the DBRM DBUTIL2 to the DB2 subsystem specified and grant the execute (run) privileges to public. Verify that it ran successfully before proceeding with this configuration.

Notes:

1. The DB2 subsystem must be Version 2 Release 3 or higher. This requirement applies to all TCP/IP servers that use DB2.
2. The DB2 subsystem must be up and running before you submit the NDBSETUP job.
3. The NDBSETUP job must be executed from a user ID with the authority to bind plans for the specified DB2 subsystem.

Step 3: Update and Install the DB2 Sample Connection Exit Routine

Note: If you will be running only one NDB server in a single address space, you may skip this step.

Because the address space running a PORTC cataloged procedure is capable of running between 1 to 20 NDB servers, changes must be made in the DB2 sample connection exit routine, DSN3SATH ASSEMBLE. These changes tell DB2 where to look for the host userid to be used for this session (the host userid was supplied by the user at NDB client invocation).

After making the necessary changes in DSN3SATH ASSEMBLE, this routine can be assembled and link edited using the DB2 supplied job DSNTIJEX in the DSNSAMP library. See DSN3SATH ASSEMBLE Modifications for NDB for the updated assembler code. For more infomation, see the *IBM DATABASE2 Administration Guide* DSNTIJEX will place a copy of the executable in the DSNEXIT library. There is one DSNEXIT library for each DB2 subsystem and it must be updated with the recommended modifications for each DB2 subsystem that will be accessed via NDB. Run DSNTIJEX once (updating the hlq for the DSNEXIT library) for each DB2 subsystem that will be accessed via NDB.

DSN3SATH ASSEMBLE Modifications for NDB

⋮

```
**********SECTION 1:  DETERMINE THE PRIMARY AUTHORIZATION ID  *********
*                                                                     *
*   IF THE INPUT AUTHID IS NULL OR BLANKS, CHANGE IT TO THE AUTHID    *
*   IN EITHER THE JCT OR THE FIELD POINTED TO BY ASCBJBNS.            *
*                                                                     *
*   THE CODE IN THIS SECTION IS AN ASSEMBLER LANGUAGE VERSION OF      *
*   THE DEFAULT IDENTIFY AUTHORIZATION EXIT.  IT IS EXECUTED ONLY     *
*   IF THE FIELD ASXBUSER IS NULL UPON RETURN FROM THE RACROUTE       *
*   SERVICE.  FOR EXAMPLE, IT DETERMINES THE PRIMARY AUTH ID FOR      *
*   ENVIRONMENTS WITH NO SECURITY SYSTEM INSTALLED AND ACTIVE.        *
*                                                                     *
***********************************************************************
         SPACE
* MOVED CHECK ON AIDLPRIM THAT WAS HERE TO PAST CHECK FOR TSO FOREGRND
         L     R7,ASCBCSCB         GET CSCB ADDRESS
         CLI   CHTRKID-CHAIN(R7),CHTSID IS IT TSO FOREGROUND ADDR SPACE
         BNE   SATH010             BRANCH IF NOT
* HERE IS NEW LOCATION OF CHECK ON AIDLPRIM
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF MOVED CODE - CHECK ON AIDLPRIM
         L     R7,ASCBJBNS         GET ADDRESS OF LOGON ID
         MVC   AIDLPRIM,0(R7)      MAKE IT THE PRIMARY AUTH ID
         B     SATH019             TO END OF THIS ROUTINE
SATH010  DS    0H                  NOT TSO, BUT BATCH OR STC SPACE
         L     R6,PSATOLD-PSA      CURRENT TCB ADDRESS
* START OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCT
         USING TCB,R6
         L     R7,TCBSENV          CURRENT ACEE ADDRESS
         USING ACEE,R7
         CLC   ACEEACEE,EYEACEE    IS THIS AN ACEE?
         BNE   SATH015             NO, GO USE JOB USER ID
         L     R14,TCBJSCB         ADDRESS OF JSCB
         USING IEZJSCB,R14
         CLC   JSCBPGMN,=CL8'PORTCLNT' IS IT NDB SERVER AS ?
         BNE   SATH015             NO, GO USE JOB USER ID
         MVC   AIDLPRIM,ACEEUSRI   MOVE USER ID INTO AIDL PRIMARY
         MVC   AIDLACEE,TCBSENV    MOVE ACEE POINTER INTO AIDL
```

```
         B     SATH020
         DROP  R6,R7,R14
SATH015  DS    0H
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCTION
         L     R7,TCBJSCB-TCB(,R6) CURRENT JSCB ADDRESS
         L     R5,JSCBJCT-IEZJSCB(,R7) CURRENT JCT ADDRESS
         LA    R5,X'10'(,R5)       ADJUST FOR CORRECT DSECT MAPPING
         MVC   AIDLPRIM(7),JCTUSER-INJMJCT(R5)  COPY JOB USER ID
         MVI   AIDLPRIM+7,BLANK    ASSURE BLANK PADDING
SATH019  DS    0H                  END OF ROUTINE
         EJECT
⋮
```

Step 4: Update the PORTS Cataloged Procedure

The PORTS procedure starts the NDB Port Manager. Update the PORTS cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTSPRC) to your system or recognized PROCLIB and modifying it to suit your local installation. Change the DD statements, as required.

Note: You must start PORTS before you start PORTC.

PORTS Cataloged Procedure (PORTSPRC)

```
//PORTS    PROC DEBUG='NOSTAE,NOSPIE/'
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB003
//*
//*        5655-HAL (C) COPYRIGHT IBM CORP. 1992, 1994.
//*        ALL RIGHTS RESERVED.
//*        US GOVERNMENT USERS RESTRICTED RIGHTS -
//*        USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*        BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*        See IBM copyright instructions
//*
//*  Change Activity:
//*
//*  $P1= PN62865 TCPV3R1 960322 LDC: Corrected options passed to C
//*
//PORTS    EXEC PGM=NDBPS,PARM='&DEBUG',TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*        The C runtime libraries should be in the system's link
//*        list or add them to the STEPLIB definition here.  If you
//*        add them to STEPLIB, they must be APF authorized.
//*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
```

```
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Step 5: Update the PORTC Cataloged Procedure

The PORTC procedure starts the NDB Port Client and the NDB Servers. Update the PORTC cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTCPRC) to your system or recognized PROCLIB and modifying it to suit your local installation.

Specifying PORTC Parameters

The PORTC parameters must be specified in order. The first four parameters are required and the last one, TRACE, is optional. For example, the following statement shows the use of all the parameters:

```
//PORTC PROC HOMEID='MVS1',USERID='NDBSRVR',DB2SSID='DB01',NUMSRV='1',TRACE='YES'
```

Parameter	Description
HOMEID	Host name that is specified on the HOSTNAME statement in the TCPIP.DATA data set. It must be able to be resolved by available host name resolution services or the HOSTS.LOCAL file.
USERID	User ID under which the NDB Server executes when not processing an NDB client request.
DB2SSID	Name of the DB2 subsystem to be accessed by NDB.
NUMSRV	Number of NDB Servers to be started. The valid range is 1 to 20.
TRACE	When tracing is turned on, it provides detailed trace of NDB Port Client and NDB Server activity for resolution by the IBM Software Support Center. You can turn tracing on by specifying: TRACE='ON' TRACE='on' TRACE='YES' TRACE='yes' All other settings, including not specifying the TRACE parameter at all, turns tracing off. Due to the volume of output that the trace generates, tracing is turned off by default.

Note: The NDB Port Client and NDB Servers that are started by PORTC cataloged procedure do not support the STOP command. Use the CANCEL command instead.

PORTC Cataloged Procedure (PORTCPRC)

```
//PORTC   PROC DEBUG='NOSTAE,NOSPIE/',
//             HOMEID='',USERID='',DB2SSID='',NUMSRV='',TRACE=''
//*
//*        TCP/IP for MVS
//*        SMP/E Distribution Name: EZAEB002
//*
//*        5655-HAL (C) Copyright IBM Corp. 1992, 1994.
//*        All rights reserved.
//*        US government users restricted rights -
//*        use, duplication or disclosure restricted
//*        by GSA ADB schedule contract with IBM Corp.
//*        See IBM copyright instructions
//*
//*  Change Activity:
```

```
//*
//*  $P1= PN62865 HTCP310 960322 LCARLOS: Corrected options passed to C
//*  $P2= MV11990 HTCP320 960322 LCARLOS: Add REGION size to invocation
//*
//PORTC    EXEC PGM=PORTCLNT,
//       PARM='&DEBUG &HOMEID &USERID NDBSRV &DB2SSID &NUMSRV &TRACE',
//         REGION=7500K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*        The C runtime libraries should be in the system's link
//*        list or add them to the STEPLIB definition here.  If you
//*        add them to STEPLIB, they must be APF authorized.
//*
//*        To use NDB Services, the DB2 load library with the
//*        suffix DSNLOAD must be in the system's link list, or
//*        added to the STEPLIB definition here.
//*        If you add it to STEPLIB, it must be APF authorized.
//*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Running Multiple PORTC Procedures

Currently the NDB Port Manager is able to manage up to 100 NDB Servers. Each PORTC procedure is able to start up to 20 NDB servers. The number of NDB Servers that can be started in a single address space depends on a number of factors, such as how large a region size can be specified and the size of the catalog work area. These factors may limit the number of NDB Servers able to start in a single address space to less than the maximum of 20.

In order to reach the desired number of NDB Servers, you can run multiple PORTC procedures. When starting a PORTC procedure, one of the parameters specified is DB2SSID. See “Specifying PORTC Parameters” on page 1082 for more information on the DBSSID parameter. This parameter indicates which DB2 subsystem all NDB servers running in that address space will access. Each started PORTC procedure may contain a different value for the DBSSID parameter; that is, when starting multiple PORTC procedures, each procedure may point to the same or different DB2 subsystems. To do this:

1. Copy your customized PORTC cataloged procedure to another data set or PDS member.
2. Change the name in the first statement (PROC statement). For example if your original PORTC procedure starts with //PORTC PROC, use //PORTC1 PROC and //PORTC2 PROC for the other procedures.
3. Ensure that:
   - HOMEID parameter settings are the same for all PORTC procedures
   - When the NUMSRV parameter settings for all the PORTC procedures are added together, the total does not exceed 100
   - The NUMSRV value for any given PORTC does not exceed 20.

4. Start the new PORTC procedures.

Step 6: Create the NDB Clients

TCP/IP for MVS provides NDB sample client code that can be compiled and run in a variety of environments. NDB clients can be created on VM, MVS, or on DOS, OS/2, AIX on RS/6000, or SUN UNIX workstations. You can find the NDB client source code in the *hlq*.SEZAINST data set, unless otherwise noted. They are written in the C language.

The process to create a client is similar in each case. Instructions for creating a client in each of the supported environments are specified. The MVS client code shipped with TCP/IP is ready to use without modification but you can modify it, if necessary, to suit your environment.

Note: To build an NDB client, you need access to the C runtime libraries and the TCP/IP RPC libraries. The TCP/IP products for the client platforms must be installed on those platforms before you build the NDB clients. In addition, for the DOS and OS/2 platforms, you must also install the TCP/IP Programmer's Toolkit.

Creating an NDB Client in the AIX Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt ndbmain.c ndbc.c ndbclt.c ndbout.c ndbpclt.c ndbx.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on AIX Version 3 Release 2 for RISC System/6000.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *TCP/IP AIX Programmer's Reference* for more information on the RPCGEN process.

Creating an NDB Client in the SUN UNIX Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt -DNOPROTO ndbmain.c ndbc.c ndbclt.c ndbout.c ndbx.c
   ndbpclt.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on SUN OS Version 4 Release 1 Modification 1.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See the *SUN Network Programming* manual for more information.

Creating an NDB Client in the OS/2 Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBCLTH	ndbclt.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Bring the following OS/2 files down to your workstation:

Program	Rename to
NDBCLDEF	ndbclnt.def
NDBOS2M	ndbos2.mak

   NDBOS2M has been stored as a binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to OS/2, make sure the transfer type used is BINARY. The OS/2 makefile (NDBOS2M) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbos2.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on OS/2 Warp Version 3.0, TCP/IP for OS/2 Version 3.0, including the TCP/IP for OS/2 Programmer's Toolkit, and IBM C Set ++ Version 2.0.
2. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new one before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *TCP/IP for OS/2 Programmer's Reference* for more information.

Creating an NDB Client in the DOS Environment

1. Bring the following C source programs down to your workstation

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

   When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h

NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

3. Bring the following DOS makefile down to your workstation:

Program	Rename to
NDBDOSM	ndbdos.mak

NDBDOSM has been stored as an binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to DOS, make sure the transfer type used is BINARY. The DOS makefile (NDBDOSM) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbdos.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on IBM DOS Version 6.0, TCP/IP for DOS Version 2.1.1.4, including the TCP/IP for DOS Programmer's Toolkit [3], Microsoft Windows Version 3.1 and Microsoft C/C++ Version 7.0.

   The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library required to run the NDB DOS client.
2. TCP/IP for DOS does not support RPC server functions because it does not have a PORTMAPPER. Therefore, if the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new one on a workstation or mainframe that does support RPC server functions before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation or mainframe:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

      ```
      rpcgen -h -o ndb.h ndb.x
      ```

   c. Transfer the resulting NDB.H file to your DOS workstation

See *TCP/IP DOS Programmer's Reference* , and the Programmer's Reference manuals for the workstation or the mainframe being used for RPCGEN for more information.

Creating an NDB Client in the VM Environment

1. Bring the following C source programs to your VM user ID:

Program	Rename to
NDBCC	NDBC C
NDBCLTC	NDBCLT C
NDBMAINC	NDBMAIN C

3. The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library that is needed for the NDB DOS client to run.

NDBOUTC	NDBOUT C
NDBPCLTC	NDBPCLT C
NDBXC	NDBX C

2. Bring the following H (header) files to your VM user ID:

Program	Rename to
NDBCLTH	NDBCLT H
NDBGLOBH	NDBGLOB H
NDBH	NDB H
NDBOUTH	NDBOUT H
NDBPCLTH	NDBPCLT H
NDBRPCFH	NDBRPCF H

3. Bring the following Assembler file to your VM user ID:

Program	Rename to
NDBVMPWA	NDBVMPW ASSEMBLE

4. Bring the following VM REXX EXEC to your VM user ID:

Program	Rename to
NDBCLBD	NDBCLBD EXEC

5. Execute the EXEC by entering:

```
ndbclbd
```

Notes:

1. This version of the NDB Sample Client code was produced and tested on VM/ESA® Version 1 Release 2 using CMS Level 9, running in XA mode and in ESA mode.
2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully compile the NDB sample client code. To do this:
   a. Bring the following X (RPC input) file to your VM user ID:

Program	Rename to
NDBXX	NDB X

   b. Issue the command:

```
rpcgen -h -o ndb h ndb x
```

See *TCP/IP for VM Programmer's Reference* for further information.

Creating an NDB Client in the MVS Environment

1. Copy the following C source programs to a PDS:

Program	Rename to
NDBCC	NDBC
NDBCLTC	NDBCLT
NDBMAINC	NDBMAIN
NDBOUTC	NDBOUT
NDBPCLTC	NDBPCLT
NDBXC	NDBX

2. Copy the following H (header) files to a PDS:

Program	Rename to
NDBCLTH	NDBCLT
NDBGLOBH	NDBGLOB

NDBH	NDB
NDBOUTH	NDBOUT
NDBPCLTH	NDBPCLT
NDBRPCFH	NDBRPCF

3. Copy the following sample JCL to a data set or PDS and customize it following the instructions found in the sample:
   NDBCLMVS
4. Execute the JCL.

Notes:

1. This version of the NDB Sample Client code was produced and tested on MVS/ESA™ Version 4 Release 2 Modification 2.
2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully build the NDB sample client. To do this:
   a. Copy the following X (RPC input) file to a data set:

Program	**Rename to**
NDBXX	NDB.X

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See the *OS/390 SecureWay Communications Server: IP Programmer's Reference*for further information.

Starting NDB

Follow these steps to start NDB:

1. Ensure that the Portmapper is up and running. For more information, see "Chapter 30. Configuring the PORTMAP Address Space" on page 1067.
2. Run the PORTSPRC procedure to start the NDB Port Manager.
3. After PORTSPRC has started, run the PORTCPRC procedure to start the NDB Port Client and NDB Servers. Specify the start parameters as required.
4. Ensure that the required DB2 subsystem is running.
5. Invoke the NDB client.

Chapter 33. Configuring the Miscellaneous (MISC) Server

Before You Configure...

Read and understand “Chapter 1. Before You Begin” on page 7. It covers important information about data set naming and search sequences.

This chapter describes how to configure the MISC server.

Understanding the MISC Server

The MISC server supports the 3 protocols described in RFCs 862, 863, and 864:
- Discard
- Echo
- Character Generator

Discard Protocol

The MISC server simply throws away any data it receives. A TCP-based server listens for TCP connections on TCP port 9. If a connection is established, the data is discarded and no response is sent. A UDP-based server listens for UDP datagrams on UDP port 9. When a datagram is received, it is discarded and no response is sent.

Echo Protocol

The MISC server returns to the originating application any data that it receives. A TCP-based server listens for TCP connections on TCP port 7. Once a connection is established, any data that is received is sent back to the originating application. A UDP-based server listens for UDP datagrams on UDP port 7. When a datagram is received, the data it contained is sent back as an answering datagram.

Character Generator Protocol

The MISC server sends a repetitive stream of character data without regard to its content. A TCP-based server listens for TCP connections on TCP port 19. When a connection is established, a stream of data is sent to the connecting application. Any data that is received is thrown away. A UDP-based server listens for UDP datagrams on port 19. When a datagram is received, an answering datagram is sent that contains a random number (between 0 and 512) of characters. The data in the received datagram is ignored.

The data that is generated follows an ordered sequence. It repeats a pattern of 94 printable ASCII characters in a ring, so that character number 0 follows character number 94.

Following is an example of the repeated pattern.

```
!"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefgh
"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghi
#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghij
$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijk
%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijkl
&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklm
'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmn
()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmno
)*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnop
*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopq
└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqr,-
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrs
-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrst
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstu
/0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuv
0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvw
123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwx
23456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxy
3456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz
456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{
56789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|
6789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}
789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~
89:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~
9:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~ !
:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~ !"
```

Configuration Process

This section describes how to configure the MISC server.

Steps to configure the MISC server:

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the MISC Server cataloged procedure (MISCSERV).

Step 1: Specify AUTOLOG and PORT Statements in PROFILE.TCPIP

To allow the MISC server to start automatically when TCPIP is initialized, include the member name of the MISC server cataloged procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP.

```
AUTOLOG
  MISCSERV
ENDAUTOLOG
```

The AUTOLOG entry in *hlq*.PROFILE.TCPIP is optional. You can choose to start the MISC server manually, when it is needed, using the START command:

```
START MISCSERV
```

Note: Because only one application can access a port, only one MISC server can execute at any given time.

The MISC server requires ports 7, 9, and 19 for both TCP and UDP. To ensure that these ports are reserved for the MISC server, verify that they are assigned to the member containing the MISC server cataloged procedure in the PORT statement in *hlq*.PROFILE.TCPIP.

```
PORT
    7 UDP MISCSERV
    7 TCP MISCSERV
    9 UDP MISCSERV
    9 TCP MISCSERV
   19 UDP MISCSERV
   19 TCP MISCSERV
```

For more information on these statements, see "AUTOLOG Statement" on page 137 and "PORT Statement" on page 229.

Step 2: Update the MISC Server Cataloged Procedure (MISCSERV)

Update the MISC server cataloged procedure by copying the sample in *hlq*.SEZAINST(MISCSERV) to your system or recognized PROCLIB and modifying the parameters and data set names to suit your local conditions.

MISC Server Cataloged Procedure (MISCSERV)

```
//MISCSERV PROC MODULE=MISCSRV,PARMS=''
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZAEC0YO
//*
//*        Licensed Materials - Program Property of IBM.
//*        This product contains "Restricted Materials of IBM"
//*        5685-061 (C) COPYRIGHT IBM CORP. 1994
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//MISCSERV EXEC PGM=&MODULE,
//           REGION=4096K,TIME=1440,
//           PARM='&PARMS'
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here. If you add
//*        them to STEPLIB, they must be APF authorized.  Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        MSMISCSR identifies an optional data set for NLS support.
//*        It specifies the MISC server message repository.
//*
//*MSMISCSR DD DSN=TCPIP.SEZAINST(MSMISCSR),DISP=SHR
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
```

```
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Specifying the MISC Server Parameters

The MISC server generates periodic messages whenever a client sends data to ports 7, 9, or 19. If this server runs continually for a long period of time, considerable amounts of spool space can be consumed. Therefore, the MISC server has all tracing turned off by default.

You can enable the trace options for any of the three MISC server protocols using the PARMS= parameter on the PROC statement of the cataloged procedure. These options will be in effect when the server starts.

TRACE
Turns on tracing for any of the specified protocols and must be followed by one or more of these three keywords:

ECho Specifies tracing for the echo protocol on port 7

DIscard
Specifies tracing for the discard protocol on port 9

CHargen
Specifies tracing for the character generator protocol on port 19

DEbug
Specifies tracing for problem determination

For example, the following statement turns tracing on for the echo and discard protocols.

```
//MISCSERV PROC MODULE=MISCSRV,PARMS='TRACE ECHO DISCARD'
```

Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent

This chapter contains information about the OS/390 UNIX Service Policy Agent and the Service Level Agreement Performance Monitoring MIB (SLAPM-MIB) Subagent.

Configuring the Service Policy Agent

The following sections provide information about the Service Policy Agent.

Overview

In most cases, current TCP/IP networks prioritize all traffic as ″best effort.″ *Service differentiation* is a mechanism to provide different service levels to different traffic types based on their requirements. *Service-level policy* is a method for controlling and regulating service differentiation. Together, service differentiation and service-level policy form an integral part of the Service Level Agreement (SLA) function that has become increasingly important as TCP/IP networks evolve.

Network administrators can use the OS/390 UNIX service policy agent (PAGENT) to define service-level policies for their users. Service-level policies provide the following benefits:

- Control of TCP throughput.
- Admission control for connection requests from clients.
- Blocking of unwanted datagrams.
- Specification of policies as active or inactive depending on the time and date.
- Prioritization of all IP traffic both within CS for OS/390 and through IP routers that use TOS settings for priority queueing and selective discards. Queued Direct Input/Output (QDIO) uses the TOS settings as the basis of its queueing structure (one queue for each priority — Network, High, Medium, and Low) and is therefore dependent on service level policy to generate the proper TOS settings for all types of IP traffic (for example, Telnet, FTP, and so on). The LINK IPAQGNET statement is used to define a gigabit ethernet link on an MPCIPA device belonging to a QDIO interface. This is the only interface supported by TCP/IP that uses QDIO.
- Providing limits for RSVP reservation parameters requested by RSVP applications, as well as limiting total active RSVP traffic flows.

PAGENT retrieves service-policy rules and statements from a policy-configuration file (/etc/pagent.conf is the default, but a specific search order is used) or from a Lightweight Directory Access Protocol (LDAP) server and installs them in one or more CS for OS/390 stacks. PAGENT sets the TOS value of outgoing IP packets, and it can be used to flush out existing policies in the stack or update them as necessary. PAGENT also logs information and error messages into the pagent log file. The location of the log file can also be specified with the PAGENT_LOG_FILE environment variable.

The service policy agent (PAGENT) runs in the OS/390 UNIX environment and is responsible for reading the policy rules and service statements from a policy configuration file or LDAP server and installing them to the TCP/IP stack. It can be used to replace existing policies or update them as necessary.

The Policy Profile Control Block (ProfileData)

There are two types of policy-related structures or objects stored in each ProfileData, the PolicyRule and the ServiceClass. The PolicyRule contains specifications for a user or group of users, such as port numbers, addresses, and protocol ID. The ServiceClass contains service information for a user or group of users, such as class-of-service, matched by the PolicyRule. There must always be a matching ServiceClass for a PolicyRule. Multiple PolicyRules can map to the same ServiceClass. One PolicyRule can refer to multiple ServiceClasses, one for each outgoing interface.

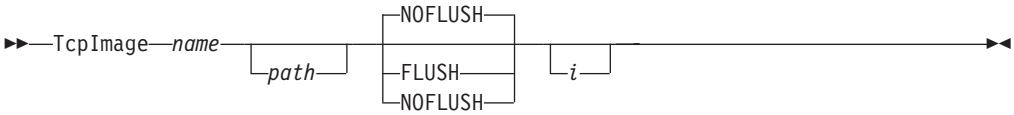
The Policy Configuration File

The policy configuration file is used when PAGENT is started. This initial configuration file can be used to point to other policy files that contain specific policies for other corresponding TCP/IP images. The statements used in the policy configuration file follow.

TcpImage Statement

The TcpImage statement specifies a TCP/IP image and its associated policy control file to be installed to that image. If a policy control file is not specified, the following control statements (if any) in this policy configuration file will be installed to that image. If no TcpImage statement is specified, any control statements will be installed to the default TCP/IP image. The parameters FLUSH or NOFLUSH (default) can be used to force flushing (deletion) of all existing policy control data in the stack.

Syntax:



Parameters:

name
The name of the TCP/IP image. The name must be 1 to 8 characters in length.

path
The path of the policy control file to be installed. If a policy control file is not specified, the following control statements (if any) in this policy configuration file will be installed.

FLUSH
Use FLUSH to delete all existing policy control data in the stack.

NOFLUSH
NOFLUSH indicates that existing policy control data in the stack should not be flushed. This is the default.

i

An integer that specifies the time interval (in seconds) for checking the creation or modification time of the configuration file(s), and for refreshing policies from the LDAP server. If a value is not specified, the default is 1800 seconds (30 minutes). The policy agent always uses this time interval to check for changes, but also monitors the configuration files(s) in real time if the -i startup option is specified.

For example, if i is set to 300, the corresponding configuration file(s) and LDAP server will be checked for changes every five minutes. If the configuration file(s) have changed within the last five minutes, they are read again. The LDAP server (if configured) is also queried again for policies. Any new, changed, or deleted policies are installed to or removed from the stack as appropriate.

Note: Dynamic monitoring is only supported for HFS files; MVS data sets are not monitored for changes.

Examples: The following example installs the policy control file */tmp/TCPCS.policy* to the *TCPCS* TCP/IP image, after flushing existing policy control data:

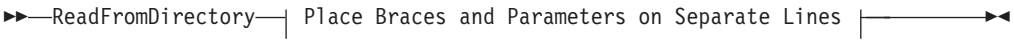
```
TcpImage TCPCS /tmp/TCPCS.policy FLUSH
```

ReadFromDirectory Statement

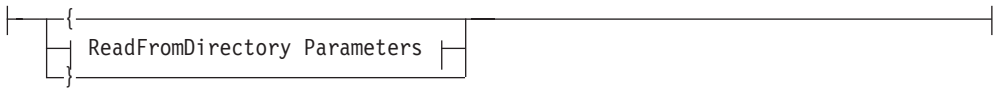
The ReadFromDirectory statement initializes PAGENT as a LDAP client. The rules are downloaded from the LDAP server, along with the rules specified in this PAGENT configuration file (the current one being used by PAGENT that contains this statement). All the rules will be installed to the appropriate TCP images.

There are three sample files that can be used to help in setting up the LDAP server and populate it with the service policies; pagent_oc.conf, pagent_at.conf, and pagent.ldif. These files reside in the /usr/lpp/tcpip/samples directory. The pagent_oc.conf file contains the policy object class definitions for the LDAP server, pagent_at.conf contains the policy attribute definitions, and pagent.ldif contains a sample of the policy rules defined for LDAP. The pagent_oc.conf and pagent_at.conf must be specified in the LDAP configuration file (for example, slapd.conf). The pagent.ldif is used when adding policy rules to LDAP. For more information on how to use LDAP and for other LDAP references, refer to *Understanding LDAP*.

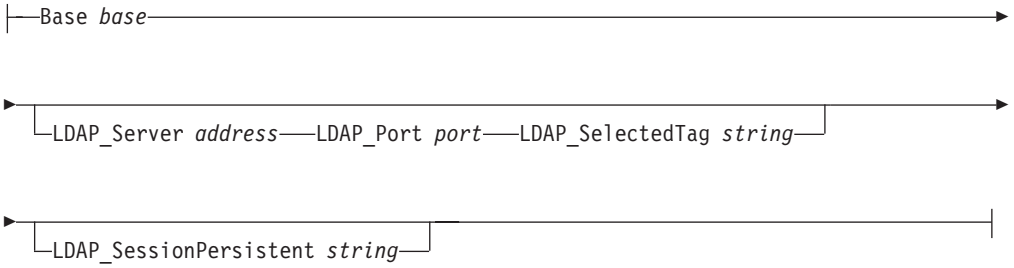
Syntax:



Place Braces and Parameters on Separate Lines:



ReadFromDirectory Parameters:



Parameters:

Base
The distinguished name of the subtree in the directory containing the policies.

LDAP_Server
The name of the server that contains policy rule definitions. The name can be specified as a character string (for example, 'ldapserver.mynetwork.com') or as an IP address (for example, 9.11.12.13). The default is the LDAP server in the local host (127.0.0.1).

LDAP_Port
The port on which the directory server is running. If not specified, the default, well-known LDAP port of 389 is used.

LDAP_SelectedTag
A string used to select a subset of the policies under the base tree. If not specified, the first machine name returned by gethostname is used.

LDAP_SessionPersistent
A string that specifies whether the LDAP session with the directory server should be kept open or closed during an update interval time. If this value is not specified, the session is closed after every query from the directory server. Valid values are yes or no. If the LDAP session update interval is small, the value of keeping the session open is greater, because it reduces the overhead of opening the session for each query.

Examples:

```
ReadFromDirectory
        {
             Ldap_server     ldapserver.mynetwork.com
             Ldap_port       9000
             Base            o=ibm,c=us
             Ldap_selectedtag      MVS1
        }
```

LogLevel Statement

The LogLevel statement specifies the level of tracing.

Syntax:

Parameters:

-i

An integer that specifies the level of logging/tracing. The supported levels are:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 8 - WARNING - Warning messages
- 16 - EVENT - Event messages
- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

Usage Notes: Specify a desired log level or a combination of levels. If this statement is absent, the default level is 15.

To combine log levels, add log level numbers. For example, to request SYSERR messages (level 1) and EVENT messages (level 16), you would request log level 17.

Examples:

```
LogLevel 511
```

SetSubnetPrioTosMask Statement

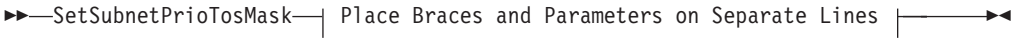
The SetSubnetPrioTosMask statement defines the TOS/priority field in the IP header type-of-service byte. It is used by the TCP/IP stack to read the TOS value and assign appropriate service to the corresponding IP packets. If this statement is not specified, TCP/IP will use the system default TOS mask and priority levels for all interfaces currently defined for IPv4 (RFC 791).

The current IPv4 TOS byte format defines the first 3 bits to be the precedent bits (for example, priority). Therefore, our default for the subnet TOS mask, if this statement is not specified, is '11100000'. In this release, only QDIO devices in S/390 can support priorities. This statement is used to set up TOS to priority mapping for those devices. QDIO supports four priority levels, 1 through 4, with 4 being the lowest priority. Following is the default mapping of these 4 priorities to the various TOS byte values:

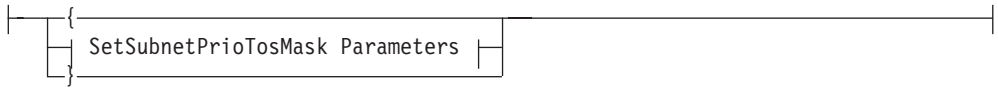
```
   TOS           Priority
00000000            4
00100000            4
01000000            3
01100000            2
10000000            1
10100000            1
11000000            1
11100000            1
```

Note that the TOS byte is also used by other network devices (for example, routers and switches) to determine the priority of a packet.

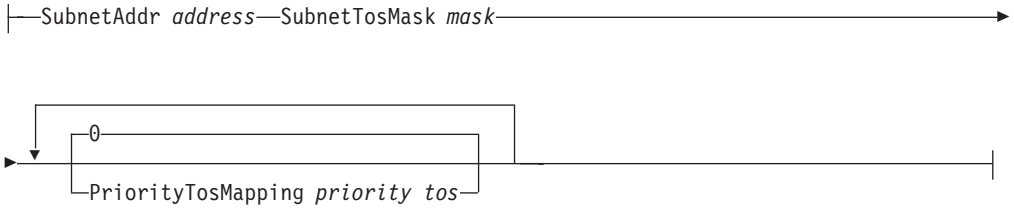
Syntax:



Place Braces and Parameters on Separate Lines:



SetSubnetPrioTosMask Parameters:



Parameters:

SubnetAddr

The local subnet interface address. A value of 0.0.0.0 indicates that the mask is the same for all interfaces.

SubnetTosMask

Eight bits, left-justified, for the TOS mask. For example, 101 would be 10100000.

PriorityTosMapping

Two values to indicate each priority level to TOS value mapping. The first value of each pair is an integer to indicate the priority level, and the second value is eight bits, left-justified, to indicate the TOS value. For example,

```
PriorityTosMapping 0 0
PriorityTosMapping 1 01000000
```

If this parameter is not specified for a TOS value, that TOS value will map to a priority value of 0.

Examples:

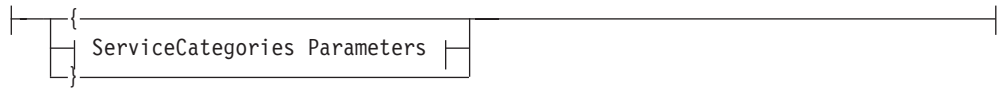
```
SetSubnetPrioTosMask
       {
            Subnetaddr      9.11.12.13
            SubnetTosMask     11100000
            PriorityTosMapping     1 00000000
            PriorityTosMapping     1 00100000
            PriorityTosMapping     2 01000000
            PriorityTosMapping     3 01100000
            PriorityTosMapping     4 11100000
        }
```

ServiceCategories Statement

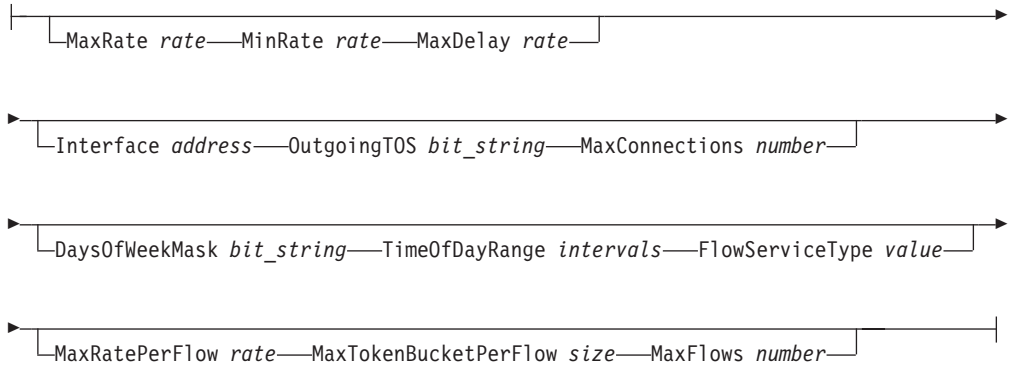
The ServiceCategories statement specifies the type of service that a flow of IP packets (for example, from a TCP connection, or UDP data) should receive end-to-end as they traverse the network. ServiceCategories can be repeated, with each having a different name so that they can be referenced later.

Syntax:

Place Braces and Parameters on Separate Lines:



ServiceCategories Parameters:



Parameters:

name
A 1 to 32 character string for the name of this service category.

MaxRate
An integer value representing the maximum rate kilobits per second (Kbps) allowed for traffic in this service class. This attribute is only valid for TCP. If not specfied or specified as 0, there will not be any enforcement of the maximum rate of a connection by the local host. If a number other than 0 is specified, each TCP connection that is mapped to this ServiceCategories will have its rate limited to this MaxRate.

MinRate
An integer value representing the minimum rate or throughput (Kbps) allowed for traffic in this service class. This attribute is only valid for TCP. If not specfied or specified as 0, there will not be any enforcement on the minimum rate of a connection by the local host. If a number other than 0 is specified, the rate for any TCP connection that is mapped to this ServiceCategories will not fall below this MinRate, unless the network is really congested and by maintaining the minimum rate the network throughput might collapse.

MaxDelay
An integer value representing the maximum delay (in milliseconds) allowed for traffic in this service class. This attribute is only valid for TCP. If not specfied or specified as 0, there will not be any enforcement on the maximum delay of a connection by the local host. If a number other than 0 is specified, the delay for any TCP connection that is mapped to this ServiceCategories will not rise above this MaxDelay, unless the network is very congested and by maintaining the maximum delay the network throughput might collapse.

Interface
The local IP subnet (for example, HOME statements) for which this service category applies. The default is all interfaces.

OutgoingTOS
Eight bits, left-justified, representing the TOS value of outbound traffic belonging to this service class. The default is 0.

MaxConnections
An integer value representing the maximum number of end-to-end connections at any instant in time. This attribute is only valid with TCP. It places a limit on the number of TCP connections mapped to this ServiceCategories that can be active at a time. If there is a request for a new TCP connection that maps to this ServiceCategories and this limit is exceeded, the connection request is rejected. The default is that there is no policy limit.

DaysOfWeekMask
A mask of seven bits representing the days in a week (Sunday through Saturday) that this service policy is active. For example, *0111110* represents weekdays. The default is all week.

TimeOfDayRange
A series of time intervals that indicate the time of day during which this service policy is active. Intervals are separated by a comma. Time starts at 0, which is right after midnight. Hours and minutes can be specified, separated by a colon. For example, the following statement would result in this policy being active after midnight until 8:30 AM, and from 5:30 PM until midnight:

```
TimeOfDayRange 0-8:30, 17:30-24
```

The default is 24 hours.

FlowServiceType
Limits the type of service being requested by RSVP applications. Valid values are ControlledLoad (the default) and Guaranteed. Guaranteed service is considered to be *greater than* ControlledLoad service. If ControlledLoad service is specified, and an application requests Guaranteed, the requested service will be downgraded to ControlledLoad. If you want to allow RSVP applications to request Guaranteed service, then specify Guaranteed for this parameter.

MaxRatePerFlow
Specifies the maximum rate in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:

- r is the token bucket rate in bytes per second
- b is the token bucket depth in bytes
- p is the peak rate in bytes per second
- m is the minimum packet size in bytes
- M is the maximum packet size (MTU) in bytes

Use this parameter to limit the *r* value of the Tspec. If an RSVP sender application requests a Tspec r value larger than this parameter, the request will be downgraded to this parameter value.

RSVP receiving applications also specify a resource specification (Rspec) when using Guaranteed service, as part of the reservation request. The Rspec consists of the following values:

- R is the rate in bytes per second
- S is the slack term in microseconds

This parameter is also used to limit the R value of the Rspec for reservation requests from RSVP receiver applications using Guaranteed service.

Note: This parameter is specified in kilobits per second, while the Tspec and Rspec use bytes per second. To arrive at a compatible specification, multiply the desired Tspec or Rspec value by 8, then divide by 1000. For example, to specify a Tspec r value of 500000 bytes per second, specify a MaxRatePerFlow value of 4000 (500000 * 8 / 1000 = 4000).

The default for this parameter is a system defined maximum.

MaxTokenBucketPerFlow
Specifies the maximum token bucket size in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:

- r is the token bucket rate in bytes per second
- b is the token bucket depth in bytes
- p is the peak rate in bytes per second
- m is the minimum packet size in bytes
- M is the maximum packet size (MTU) in bytes

This parameter is used to limit the *b* value of the Tspec. If an RSVP sender application requests a Tspec b value larger than this parameter, the request will be downgraded to this parameter value.

Note: This parameter is specified in kilobits, while the Tspec uses bytes. To arrive at a compatible specification, multiply the desired Tspec value by 8, then divide by 1000. For example, to specify a Tspec b value of 75000 bytes, specify a MaxTokenBucketPerFlow value of 600 (75000 * 8 / 1000 = 600).

The default for this parameter is a system defined maximum.

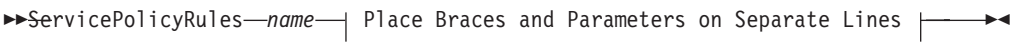
MaxFlows
Specifies the maximum number of reserved flows allowed for RSVP applications. The default is no limit on the number of reserved flows.

Examples: For an example of the ServiceCategories statement, see /usr/lpp/tcpip/samples/pagent.conf.

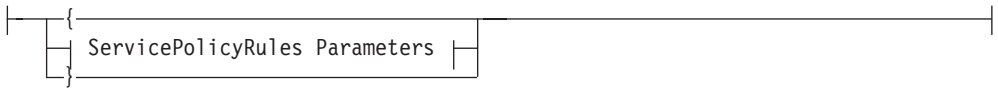
ServicePolicyRules Statement

The ServicePolicyRules statement specifies characteristics of IP packets that are used to map to a corresponding service category; it defines a set of IP datagrams that should receive a particular service.

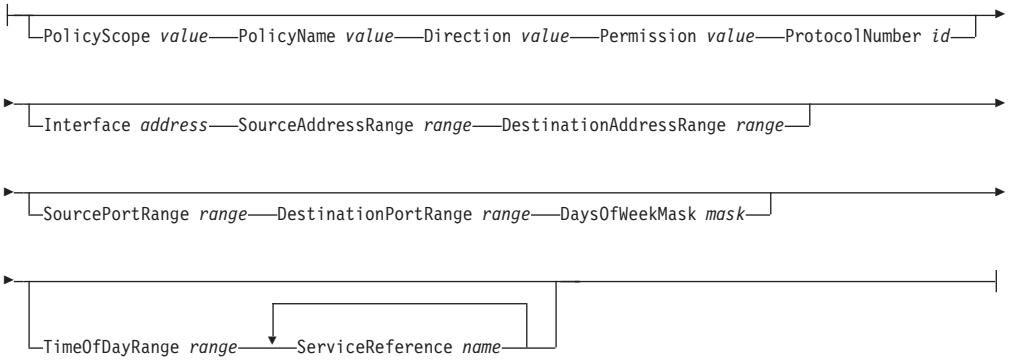
Syntax:



Place Braces and Parameters on Separate Lines:



ServicePolicyRules Parameters:



Parameters:

name
A 1 to 32 character string for the name of this policy rule.

PolicyScope
Indicates to what traffic this policy rule applies. Valid values are *DataTraffic*, *RSVP*, and *Both*. The default is *DataTraffic*. When RSVP (Resource reSerVation Protocol, a network protocol running on top of IP) is specified, this policy only applies to data that are specifically reserved by using RSVP. When Datatraffic is specified, the policy applies to all other non-RSVP data.

PolicyName
An administrative name for this policy. This name can be used to group service policy rules. The name can be from 1 to 32 characters long.

Direction
Indicates the direction of traffic for which this policy rule applies. Valid values are *Incoming*, *Outgoing*, and *Both*. The default is *Outgoing*.

It is important to note that policies are applied to TCP on a connection basis, whereas they are applied to UDP/RAW on a per-packet basis. Therefore, the Direction attribute is also mapped accordingly. More specifically, if a policy is

defined for TCP, the Direction attribute will apply to the direction of the connection (inbound if the local 390 host is to receive the connection request, such as incoming TCP SYN segments). If a policy is defined for UDP/RAW, Direction will apply to individual packets.

Permission
Indicates whether packets belonging to this policy rule should be discarded or allowed to proceed. Valid values are *Allowed* and *Blocked*. The default is *Allowed*.

ProtocolNumber
This is a one byte field in the IP header to identify the protocol running on top of IP. Common protocols are UDP and TCP. For UDP, TCP, and RAW, this field can be specified with these names. For others, a number has to be specified (for example, 1 for ping). The default is all protocols.

Interface
The local IP subnet for which this policy rule applies. The default is all interfaces.

SourceAddressRange
The local IP address range. This field consists of two addresses, separated by a space, where the first address is less than or equal to the second address. The default is 0, which is all inclusive.

SourceAddressRange is the address range of addresses that are local to the 390 host (for example, defined via HOME statements in the TCP/IP configuration).

DestinationAddressRange
The remote IP address range. This field consists of two addresses, separated by a space, where the first address is less than or equal to the second address. The default is 0, which is all inclusive.

DestinationAddressRange is the address range of the remote hosts that are communicating with the local 390 host.

SourcePortRange
The local port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive.

SourcePortRange contains the address range of the remote hosts that are communicating with the local 390 host.

DestinationPortRange
The remote port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive.

DestinationPortRange contains the address range of the remote hosts that are communicating with the local 390 host.

DaysOfWeekMask
A mask of seven bits representing the days in a week (Sunday through Saturday) that this policy rule is active. For example, *0111110* represents weekdays. The default is all week.

TimeOfDayRange
A series of time intervals that indicate the time of day during which this policy

rule is active. Intervals are separated by a comma. Time starts at 0, which is right after midnight. Hours and minutes can be specified, separated by a colon. For example, the following statement would result in this policy rule being active after midnight until 8:30 AM, and from 5:30 PM until midnight:

```
TimeOfDayRange 0-8:30, 17:30-24
```

The default is 24 hours.

ServiceReference

Indicates the name of a service category from a service category statement (for example, interactive) that this policy rule uses. One or more service category names can be specified to associate this policy rule with different interfaces or different service policies depending, for example, on the time when each of those service policies are active.

Examples: For an example of the ServicePolicyRules statement, see /usr/lpp/tcpip/samples/pagent.conf.

Starting the Service Policy Agent

You can start the service policy agent from the OS/390 shell or as a started task.

Although the /etc/pagent.conf is the default configuration file, a specific search order is used when starting the service policy agent. The following order is used:

- File or data set specified with the -c startup option
- File or data set specified with the PAGENT_CONFIG_FILE environment variable
- /etc/pagent.conf
- hlq.PAGENT.CONF

At initialization, PAGENT creates an HFS file called /tmp/*tcpname*.Pagent.tmp. This occurs for every TCP/IP stack defined on a TcpImage statement.

In this HFS file, tcpname is the name of a TCP/IP stack from a TcpImage statement. During TCP/IP stack initialization, the TCP/IP stack will attempt to modify a file by this name to notify the PAGENT that the stack has been reactivated. This causes the PAGENT to automatically attempt to reinstall the existing policies to this stack.

Note: This process only works if the -i startup option is specified. This directs PAGENT to monitor files in real time for changes.

To insure that only one service policy agent is started, the service policy agent uses the following enqueue:

- Enqueue name is TCP_TCPI
- Resource name is TCPIP.PAGENT

Starting Pagent from the OS/390 Shell

The pagent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin.

-c The -c option allows a policy configuration file name to be specified. If it is not specified, the configuration file is located using the search order.

-d The -d option can be used to get more tracing information during pagent execution. When -d is specified, all trace messages are issued to stdout. The trace messages are also logged in the policy agent log file. If -d is not used, log messages are written to the policy agent log file as specified by the LogLevel configuration statement. The log file should be the first place checked for error messages.

-i When specified, the policy agent monitors its local files in real time for changes. Local files include all configuration files and the temporary files used by the TCP/IP stacks to inform the policy agent of stack restarts. The time interval configured on the TcpImage statement is also used to monitor configuration files and the LDAP server for updates. Use of the -i option provides the following benefits:

- More timely updating of policy statements when a configuration file is changed
- Notification when a stack is restarted, so policies can be reinstalled on that stack.

Notes:

1. You can cause an immediate refresh of policy from the LDAP server by changing the configuration file, which causes the file to be reread. If the file is configured to read policy from the LDAP server, PAGENT does so at that time.
2. Dynamic monitoring is only supported for HFS files; MVS data sets are not monitored for changes.
3. Notification of stack restarts is only supported when -i is specified.

Starting Pagent as a Started Task

Use the S PAGENT command on an MVS console or SDSF to start pagent. A sample procedure is shipped in member EZAPAGSP in *hlq*.SEZAINST. All of the information regarding default locations for the configuration and log files is the same as for starting from the OS/390 shell. Following is a copy of the sample procedure:

```
//PAGENT   PROC
//*
//* SecureWay Communications Server IP
//* SMP/E distribution name: EZAPAGSP
//*
//* 5647-A01 (C) Copyright IBM Corp. 1999.
//* Licensed Materials - Property of IBM
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
//*            etc/pagent3.conf'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   PAGENT_CONFIG_FILE=/etc/pagent2.conf
//*   PAGENT_LOG_FILE=/tmp/pagent2.log
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.PAGENT.ENV(PAGENT),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/pagent.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the PAGENT_LOG_FILE environment variable, and defaults to
//* /tmp/pagent.log.  When the -d parameter is specified, however,
//* output is also written to stdout.
//*
//SYSPRINT DD SYSOUT=*
```

```
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Stopping the Service Policy Agent

Pagent can be stopped using the cancel command (C PAGENT) or using the kill command in the OS/390 shell. The following kill command with the TERM signal will enable pagent to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is the pagent process ID. The pagent process ID can be obtained using the following OS/390 UNIX command:

```
ps -A
```

It can also be obtained from the /tmp/pagent.pid file.

The /tmp/pagent/pid file is a temporary file created by the Policy Agent. It contains the process id of the current (or last) invocation of the Policy Agent.

Configuring the SLA Subagent

The following sections describe the Service Level Agreement Performance Monitoring MIB (SLA) subagent, including configuration information.

SLA Subagent Performance Monitoring

The SLA Subagent provides information about service policies and performance data for applications which are mapped to those policies via two tables:

slapmPolicyStatsTable
: Provides information about defined service policies and aggregate performance data for mapped applications.

slapmSubcomponentTable
: Provides information about individual TCP or UDP applications and application-specific performance data.

The SLA Subagent also supports performance monitoring via the slapmPolicyMonitorTable object. Entries are created in the monitor table to establish the desired criteria for monitoring. The following levels of monitoring are provided:

Aggregate
: Monitoring is performed based on the aggregate of all TCP or UDP applications that are mapped to one or more service policies.

Subcomponent
: Monitoring is performed based on a single TCP or UDP application.

Three types of monitoring are provided for measuring application performance:

MinRate
: The current input/output rates of the application(s) are compared to threshold values established in the monitor table entry. If the current rates are less than the threshold, an SNMP trap is sent if traps are enabled.

MaxRate
The current input/output rates of the application(s) are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

MaxDelay
The current delay rates of the application(s) are calculated by using TCP round trip time (RTT). For aggregate monitoring, the RTT of all TCP applications are averaged. The delay rates are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

Note: MaxDelay monitoring is only available for TCP applications.

Refer to the draft RFC for the SLAPM MIB in the sample file slapm.text in the /usr/lpp/tcpip/samples directory for more details about how to make the various monitoring calculations.

When SNMP traps are enabled, and a *not achieved* trap is sent as described above, a corresponding *okay* trap is sent when the traffic once again conforms to the boundaries established in the monitor table entry.

For example, you can establish MaxDelay monitoring and use a max delay low value of 50 and a max delay high value of 75. If the RTT of the application(s) rises above 75, a *not achieved* trap is sent. If the RTT then drops below 50, an *okay* trap is sent to indicate the problem has been resolved. However, if the application(s) end before conforming to the established boundaries, an okay trap may not be sent naturally. For application level traps, the okay trap is never sent because the corresponding subcomponent table entry gets deleted when the application(s) end, which also removes the application level monitoring.

For aggregate traps, when the application(s) end, MaxRate and MaxDelay okay traps will be sent, because a value of 0 for each of these should fall below the minimum values established in the monitor table entry. Conversely, for MinRate monitoring, an aggregate okay trap is not sent, because a value of 0 will never be greater than the maximum value established in the monitor table.

In addition to the traps used to measure application performance, two additional traps are used to monitor table administration:

Policy Deleted
Trap is sent when an entry is deleted from the slapmPolicyStatsTable.

Monitor Deleted
Trap is sent when an entry is deleted from the slapmPolicyMonitorTable.

Creating Monitor Table Entries and Enabling SNMP Traps

Several MIB objects are used when establishing monitor table entries and for configuring whether and how often traps are sent. First, to establish monitor table entries, set the following MIB object variables.

Note: Because most of these objects have default values, you might be able to achieve the desired monitoring using only a subset of the objects.

MIB Object	Action
slapmPolicyMonitorControl	Controls what levels and types of monitoring are in effect.

slapmPolicyMonitorInterval Sets the interval for calculating input/output and delay rates and checking those values against the monitor table thresholds.

slapmPolicyMonitorMinRateLow, slapmPolicyMonitorMinRateHigh, slapmPolicyMonitorMaxRateLow, slapmPolicyMonitorMaxRateHigh, slapmPolicyMonitorMaxDelayLow, slapmPolicyMonitorMaxDelayHigh
Establishes the threshold values for MinRate, MaxRate, and MaxDelay monitoring. The min/max rates are in units of kilobits per second, and the max delay is in units of milliseconds.

slapmPolicyMonitorRowStatus
Controls the status of a monitor table entry, for instance whether or not the entry is active.

In addition, the following MIB objects are used to control the generation of traps

MIB Object
Action

slapmPolicyTrapEnable
Enables or suppresses generation of Policy Deleted and Monitor Deleted traps.

slapmPolicyTrapFilter
Establishes the number of times a given MinRate, MaxRate, or MaxDelay event must be encountered before a trap is generated.

slapmPolicyPurgeTime
Establishes a timeout value for Policy Deleted traps. After a service policy is deleted, the amount of time indicated by this object must expire before a Policy Deleted trap is generated.

slapmPolicyMonitorControl
Controls whether or not aggregate and/or subcomponent traps are enabled.

Creating the Monitor Table Index

When you create monitor table entries, specify the appropriate index value. The index is composed of the following

- slapmPolicyMonitorOwnerIndex
- slapmPolicyMonitorSystemAddress
- slapmPolicyMonitorPolicyName
- slapmPolicyMonitorTrafficProfileName

Each of these values is expressed in the format:

```
length.character.character...
```

where character is in ASCII decimal form.

For example, the value ″u1″ is expressed as ″2.117.31″. The PolicyName part of the index matches the value specified on the PolicyName keyword on the ServicePolicyRules statement for the associated policy definition, while the TrafficProfileName matches the rule name specified on the ServicePolicyRules statement.

For example, assume the following service policy is defined:

```
ServiceCategories   ElaineCat
     {
          Priority       1
          MaxRatePerFlow   640
          MaxTokenBucketPerFlow 440
          Maxdelay       1
     }
ServicePolicyRules  ElainePol
     {
          Direction     both
          PolicyScope   rsvp
          ProtocolNumber     TCP
          SourcePortRange    8000
          ServiceReference   ElaineCat
          PolicyName         eam
     }
```

In the above example, the PolicyName is ″eam″ and the TrafficProfileName is ″ElainePol″. To assist you in creating the index for the monitor table entries, note that the index value used in the slapmPolicyStatsTable entries consists of the last three values used in the monitor table index, namely the SystemAddress, PolicyName, and TrafficProfileName. So, you can walk the slapmPolicyStatsTable using the following command:

```
osnmp -v walk slapmPolicyStatsTable
```

Then, cut-and-paste the index value from the PolicyStatsTable, and add an OwnerIndex of your choosing at the beginning of the index. For the above example, the complete index using an OwnerIndex of ″u1″ is:

```
2.117.31.0.3.101.97.109.9.69.108.97.105.110.101.80.111.108
|         | |             |
|         | |             +- length + "ElainePol" (TrafficProfileName)
|         | +-------------- length + "eam" (PolicyName)
|         +---------------- no SystemAddress (SystemAddress)
+------------------------- length + "u1" (OwnerIndex)
```

Monitor Table Examples:

Note: If you are going to change any of the monitor table object values for an existing table entry or row, you must take the row out of service to make the changes. To do this, set the value of slapmPolicyMonitorRowStatus to 2.

After your changes are made, set the row status to a value of 1 to put it back in service.

Two of the monitor table objects, monitor control and monitor status fields, are important in setting up the entries and understanding why traps are generated. Both of these fields have the SNMP data type BITS, which means they are bit strings, where bit 0 is the low order bit. Any combination of bits can be set into these objects.

Table 48 shows the meaning of the various bits:

Table 48. Monitor Control and Monitor Status Object Bit Values

MonitorControl Object	xx54 3210
0 - monitor MinRate	0000 0001 = 01
1 - monitor MaxRate	0000 0010 = 02
2 - monitor MaxDelay	0000 0100 = 04

Table 48. Monitor Control and Monitor Status Object Bit Values (continued)

3 - enable aggregate traps	0000 1000 = 08
4 - enable subcomponent traps	0001 0000 = 10
5 - monitor subcomponent	0010 0000 = 20
MonitorStatus Object	xx98 7654 3210
0 - slaMinInRateNotAchieved	0000 0000 0001 = 001
1 - slaMaxInRateExceeded	0000 0000 0010 = 002
2 - slaMaxDelayExceeded	0000 0000 0100 = 004
3 - slaMinOutRateNotAchieved	0000 0000 1000 = 008
4 - slaMaxOutRateExceeded	0000 0001 0000 = 010
5 - monitorMinInRateNotAchieved	0000 0010 0000 = 020
6 - monitorMaxInRateExceeded	0000 0100 0000 = 040
7 - monitorMaxDelayExceeded	0000 1000 0000 = 080
8 - monitorMinOutRateNotAchieved	0001 0000 0000 = 100
9 - monitorMaxOutRateExceeded	0010 0000 0000 = 200

The following examples show how to create monitor table entries to monitor at various levels and types. You can also create other combinations using the monitor control object. This example assumes SNMP version 1 security and no snmpd.conf file.

First, enable traps. The snmptrap.dest file should contain the IP address and protocol of an entity to receive traps.

```
/etc/snmptrap.dest   contains: 9.67.191.5    UDP
/etc/pw.src          contains: public 0.0.0.0  0.0.0.0
```

In this example, the osnmp command running in the background is used to receive traps.

```
  osnmp set snmpEnableAuthenTraps.0 1
  osnmp set slapmPolicyTrapEnable.0 1
  osnmp set slapmPolicyTrapFilter.0 1
  osnmp set slapmPolicyPurgeTime.0 60
  osnmp trap > /tmp/trap.output &
```

To monitor for MinRate at an aggregate level:

```
osnmp set slapmPolicyMonitorControl.index \'00000009\'h
  where  9 is aggregate monitor and trap for MinRate
     or choose 1 for aggregate monitor only for MinRate

osnmp set slapmPolicyMonitorMinRateLow.index  l
osnmp set slapmPolicyMonitorMinRateHigh.index h
  where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
    to start monitoring
```

If the MinRate occurs, look at the monitor status object in the trap, or:

```
osnmp get slapmPolicyMonitorStatus.index
   should be 1 if inbound MinRate not achieved
     or 8 if outbound MinRate not achieved
```

To monitor for MaxRate at an aggregate level:

```
osnmp set slapmPolicyMonitorControl.index \'0000000a\'h
   where  a is aggregate monitor and trap for MaxRate
      or choose 2 for aggregate monitor only for MaxRate

osnmp set slapmPolicyMonitorMaxRateLow.index  l
osnmp set slapmPolicyMonitorMaxRateHigh.index h
   where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
   to start monitoring
```

If the MaxRate occurs, look at the monitor status object in the trap, or:

```
 osnmp get slapmPolicyMonitorStatus.index
   should be 2 if inbound MaxRate exceeded
     or 10 if outbound MaxRate exceeded
```

To monitor for MaxDelay at an aggregate level:

```
osnmp set slapmPolicyMonitorControl.index \'0000000c\'h
   where  c is aggregate monitor and trap for MaxDelay
     or choose 4 for aggregate monitor only for MaxDelay

osnmp set slapmPolicyMonitorMaxDelayLow.index  l
osnmp set slapmPolicyMonitorMaxDelayHigh.index h
   where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
   to start monitoring
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPolicyMonitorStatus.index
  should be 4 if MaxDelay exceeded
```

To monitor for MinRate at an application level:

```
osnmp set slapmPolicyMonitorControl.index \'00000031\'h
  where 31 is subcomponent monitor and trap for MinRate
    or choose 21 for subcomponent monitor only for MinRate

osnmp set slapmPolicyMonitorMinRateLow.index  l
osnmp set slapmPolicyMonitorMinRateHigh.index h
  where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
    to start monitoring
```

If the MinRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPolicyMonitorStatus.index
  should be 20 if inbound MinRate not achieved
    or 100 if outbound MinRate not achieved
```

To monitor for MaxRate at an application level:

```
osnmp set slapmPolicyMonitorControl.index \'00000032\'h
  where 32 is subcomponent monitor and trap for MaxRate
    or choose 22 for subcomponent monitor only for MaxRate

osnmp set slapmPolicyMonitorMaxRateLow.index  l
osnmp set slapmPolicyMonitorMaxRateHigh.index h
   where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
   to start monitoring
```

If the MaxRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPolicyMonitorStatus.index
  should be 40 if inbound MaxRate exceeded
    or 200 if outbound MaxRate exceeded
```

To monitor for MaxDelay at an application level:

```
osnmp set slapmPolicyMonitorControl.index \'00000034\'h
  where 34 is subcomponent monitor and trap for MaxDelay
    or choose 24 for subcomponent monitor only for MaxDelay

osnmp set slapmPolicyMonitorMaxDelayLow.index  l
osnmp set slapmPolicyMonitorMaxDelayHigh.index h
  where l is the lower boundary and h is the upper boundary

osnmp set slapmPolicyMonitorRowStatus.index 1
   to start monitoring
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPolicyMonitorStatus.index
  should be 80 if MaxDelay exceeded
```

Starting the SLA Subagent

The SLA subagent provides information about service policies and performance data about applications mapped to those policies.

Warning

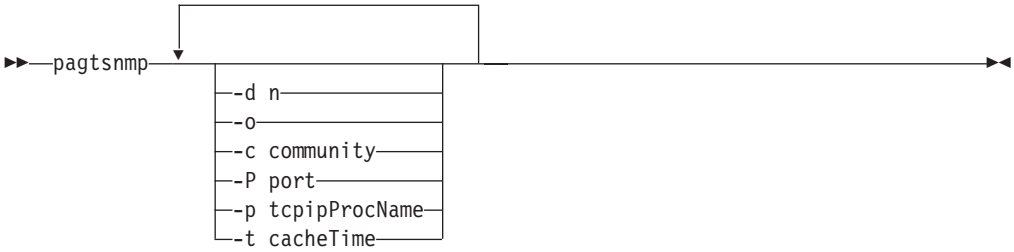
For CS for OS/390 V2R8, the SLA subagent objects are experimental and occupy the experimental portion of the MIB tree. The MIB objects may change in future releases.

You can start the SLA Subagent from the OS/390 shell or as a started task.

Starting the SLA Subagent from the OS/390 Shell

The SLA Subagent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your path statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

Syntax



Parameters

-d n
Specifies that the subagent should run in debugging mode. The following modes are supported:

1 Internal debugging messages are written.

2 Internal debugging messages, output from the ioctls issued to the stack, and DPIdebug() level 2 output are written.

Output from the -d parameter is written to syslogd or stdout depending on the -o parameter. The debug level can be dynamically changed via a MODIFY command.

-o Specifies that debug output should be written to stdout. The default is to write to syslogd.

-c A character string of up to 32 characters used as the SNMPv1 community name (or password) in establishing contact with the SNMP Agent. For pagtsnmp to communicate with the CS for OS/390 SNMP Agent, the community name specified on the -c startup option must match one that is defined in a data set configured to the SNMP Agent (or defaulted) on the -c parameter when the SNMP Agent is started. The default value is ″public″.

-P port
A port number between 1 and 65535 used in establishing communication with the SNMP Agent. For pagtsnmp to communicate with the CS for OS/390 SNMP Agent, the port number specified must match the port number specified (or defaulted) on the -p parameter when the SNMP Agent is started. The default value is 161.

-p tcpipProcName
The tcpipProcName is an 8-byte procedure name that is used to start TCP/IP. If this parameter is not specified, pagtsnmp uses the standard resolver configuration search order to determine this parameter.

-t cacheTime
Amount of time in seconds to elapse before rebuilding the MIB object tables.

Starting the SLA Subagent as a Started Task

Use the S PAGTSNMP command on an MVS console or SDSF to start the SLA subagent. A sample procedure is shipped in member EZAPAGSNMP in *hlq*.SEZAINST. Following is a copy of the sample procedure:

```
//PAGTSNMP PROC
//*
//* SecureWay Communications Server IP
//* SMP/E distribution name: EZAPAGSN
//*
//* 5647-A01 (C) Copyright IBM Corp. 1999.
//* Licensed Materials - Property of IBM
//*
//PAGTSNMP EXEC PGM=PAGTSNMP,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c p
//*            ublic -P 1234'
//*
//* Provide environment variables to run with the desired stack.  As
//* an example, the data set or file specified by STDENV could
//* contain:
```

```
//*
//*   RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA)'
//*
//* For information on the above environment variable, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.PAGTSNMP.ENV(PAGTSNMP),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/pagtsnmp.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, PAGTSNMP doesn't write output to stdout or stderr.
//* Instead, output is written to syslogd.  When the -o parameter
//* is specified, however, output is written to stdout instead of
//* syslogd.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Stopping the SLA Subagent

The SLA Subagent can be stopped using the stop command (P PAGTSNMP) or using the kill command in the OS/390 shell. The following kill command with the TERM signal will enable the SLA Subagent to clean up resources properly before terminating itself:

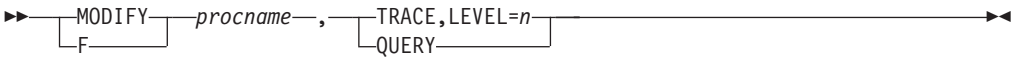
```
kill -s TERM pid
```

where *pid* is the pagtsnmp process ID. The pagtsnmp process ID can be obtained using the following OS/390 UNIX command:

```
ps -A
```

Controlling the SLA Subagent with the Modify Command

You can control the SLA subagent functions from the operator's console using the MODIFY command. The following is the syntax and valid parameters.



Parameter	Description
TRACE,LEVEL	Changes the SLA subagent debug level. *n* is the desired debug level. Specifying a level of 0 disables debug tracing.
QUERY	Displays the current SLA subagent debug level in effect.

Security Information

Because the Service Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the service policy agent.

The following security product profile name must be defined:

```
MVS.SERVMGR.PAGENT
```

Listed below are sample commands used to create this profile name and permit users to it:

```
RDEFINE OPERCMDS (MVS.SERVMGR.PAGENT) UACC(NONE)
PERMIT MVS.SERVMGR.PAGENT ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Chapter 35. Configuring the RSVP Agent

This chapter contains information about the following topics:

- Description of RSVP
- Configuring RSVP
- Starting RSVP
- Stoppping RSVP
- Security information

Description of RSVP

In most cases, current TCP/IP networks prioritize all traffic as *best effort*. Service differentiation is a mechanism to provide different service levels to different traffic types based on their requirements. Quality of Service (QoS) is the overall service that a user or application receives from a network, in terms of throughput, delay, and so on. The following types of service differentiation have been defined for TCP/IP networks:

- Integrated Services provides end-to-end QoS to an application, using the methodology of resource reservation along the data path from a receiver to a sender
- Differentiated Services provides QoS to broad classes of traffic or users, for example all FTP traffic to a given subnet

Resource ReSerVation Protocol (RSVP) is a server, or agent, that provides a mechanism to reserve resources in support of Integrated Services. The OS/390 UNIX RSVP agent provides the following services on behalf of applications that want to use Integrated Services:

- An RSVP API (RAPI) that allows applications to explicitly request RSVP services. Using RAPI, applications indicate their intent to send or receive data, describe the characteristics of the data traffic and request that RSVP reserve resources along the data path to provide a given QoS to one or more traffic flows. For more information about RAPI, refer to *OS/390 SecureWay Communications Server: IP Programmer's Reference*.
- Mapping of IP TOS settings to RSVP traffic, using service policies defined for RSVP.
- Establishment of resource reservations on ATM interfaces by use of reserved SVC connections.
- Support for VIPA addresses as well as real IP addresses.
- Communication with other RSVP agents on hosts/routers in the network to communicate application resource reservation requests.

Network administrators can use the OS/390 UNIX service policy agent to define RSVP-specific policies. These policies can be used to limit the parameters of application-requested resource reservations, provide TOS mappings for RSVP traffic, and limit the number of traffic flows that can use RSVP services simultaneously. For additional information, see "Chapter 34. Configuring the OS/390 UNIX Service Policy Agent and SLA Subagent" on page 1095.

RSVP is designed to be implemented on both end systems (hosts) and routers. Different functions are provided by RSVP in these two environments. The OS/390 RSVP agent is supported as a host RSVP implementation only. It can communicate

with router RSVP implementations, but is not itself supported as such. For more information about RSVP, refer to RFC 2205.

Configuring the RSVP Agent

Follow these steps to configure the RSVP agent:

1. Use the configuration file to specify RSVP agent operational parameters.
2. Start the RSVP agent.

The RSVP Configuration File

The RSVP Agent uses the following search order to locate the configuration file (highest priority is listed first):

- HFS file or MVS data set specified by the -c startup option. The syntax for an HFS file is '/dir/file' and the syntax for an MVS data set is ″//'MVS.DATASET.NAME'″.
- HFS file or MVS data set specified with the RSVPD_CONFIG_FILE environment variable.
- /etc/rsvpd.conf HFS file.
- 'hlq.RSVPD.CONF' MVS data set.

Note: If this file is not present, RSVP is enabled on all network interfaces with default parameters.

LogLevel Statement

The LogLevel statement specifies the level of tracing.

Syntax

Parameters

-i

An integer that specifies the level of logging/tracing. The supported levels are:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 8 - WARNING - Warning messages
- 16 - EVENT - Event messages
- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

Usage Notes

Specify a desired log level or a combination of levels. If this statement is absent, the default level is 15.

To combine log levels, add log level numbers. For example, to request SYSERR messages (level 1) and EVENT messages (level 16), you would request log level 17.

Examples

```
LogLevel 511
```

TcpImage Statement

The TcpImage statement identifies the name of the stack to which the RSVP agent should establish affinity.

If this statement is absent, the RSVP agent establishes affinity with the default stack.

Syntax

Parameters

name
The name of the TCP/IP image. The name must be 1 to 8 characters.

Examples

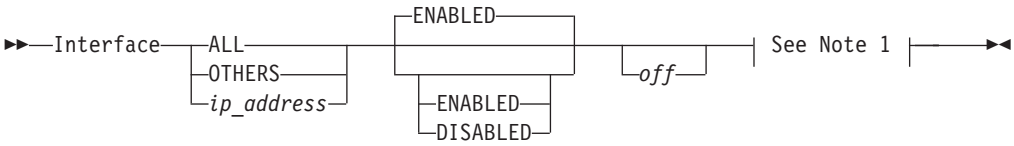
```
TcpImage TCPCS2
```

Interface Statement

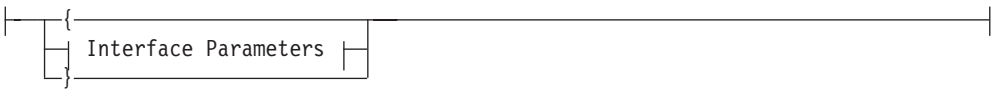
The Interface statement makes available to the RSVP agent one or more of the network interfaces of the local host.

If this statement is absent, none of the network interfaces are available to the RSVP agent.

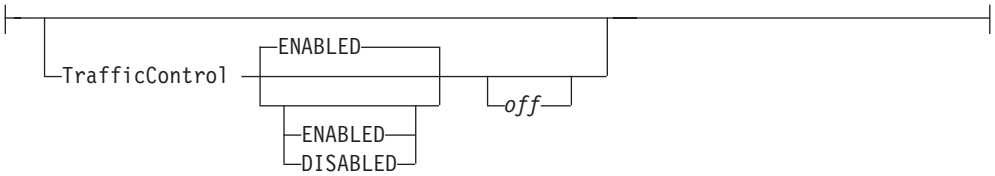
Syntax



Note 1: Place Braces and Parameters on Separate Lines:



Interface Parameters:



Parameters

IP_address
The IP address (dotted decimal format) of the interface. You can choose a specific interface IP address such as *all*, which means all configured interfaces (currently configured on the HOME statement or dynamically added in the future), or *others* which means all interfaces except those previously configured.

In the following example, all interfaces except 9.10.11.12 would be enabled.

```
Interface 9.10.11.12 Disabled
Interface others Enabled
```

Enabled
Specifies that RSVP should use this interface.

Disabled
Specifies that RSVP should not use this interface.

Off
Specifies to ignore this statement.

TrafficControl

Specifies whether or not traffic control is in effect. When traffic control is disabled, the RSVP agent does not install any filters (resource reservations). If off is specified, the traffic control specification portion of the Interface statement will be ignored.

Examples

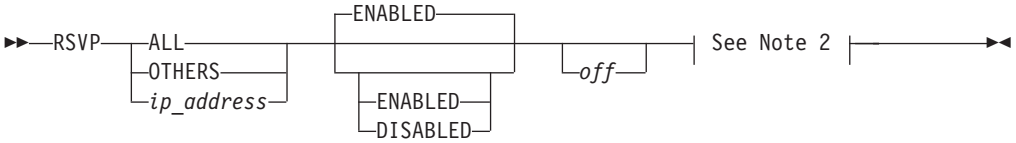
```
Interface 9.23.78.13
{
  trafficcontrol enabled
}
interface others disabled
interface all
```

RSVP Statement

The RSVP statement enables RSVP processing on one or more of the network interfaces of the local host.

If this statement is absent, RSVP processing is disabled on all network interfaces.

Syntax



Note 2: Put Braces and Parameters on Separate Lines:

RSVP Parameters:

Parameters

IP_addressstart here
The IP address (dotted decimal format) of the interface. You can choose a specific interface IP address such as *all*, which means all configured interfaces (currently configured on the HOME statement or dynamically added in the future), or *others* which means all interfaces except those previously configured.

In the following example, all interfaces except 9.10.11.12 would be enabled.

```
Interface 9.10.11.12 Disabled
Interface others Enabled
```

Enabled
Specifies that RSVP processing should use this interface.

Disabled
Specifies that RSVP processing should not use this interface.

Off
Specifies to ignore this statement.

MaxFlows
Specifies the maximum number of data flows.

i An integer defining the maximum number of data flows to be allowed using this interface. The default is 32.

Examples

```
rsvp 87.13.112.6

{
maxflows 100
}
rsvp others
rsvp all
```

Starting the RSVP Agent

You can start the RSVP Agent from the OS/390 shell or as a started task.

Although the /etc/rsvpd.conf is the default configuration file, a specific search order is used when starting the RSVP agent. The following order is used:

- File or data set specified with the -c startup option
- File or data set specified with the RSVPD_CONFIG_FILE environment variable
- /etc/rsvpd.conf
- hlq.RSVPD.CONF

Note: RSVP creates the temporary file /tmp/rsvp.pid.imagename, which contains the process ID of the current (or last) invocation of the RSVP agent. The imagename in the file is the TCP/IP stack name, for example /tmp/rsvpd.pid.TCPCS2.

To insure that only one RSVP Agent is started per TCP/IP stack, the RSVP Agent uses the following enqueue:

- Enqueue name is TCP_TCPI
- Resource name is TCPIP.RSVPD.stackname

Starting RSVP from the OS/390 Shell

The rsvp executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your path statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

-c The -c option allows a RSVP Agent configuration file to be specified. If it is not specified, the configuration file is located using the search order.

Starting RSVP as a Started Task

Use the S RSVPD command on an MVS console or SDSF to start RSVP. A sample procedure is shipped in member EZARSVPP in *hlq*.SEZAINST. All of the information regarding default locations for the configuration and log files is the same as for starting from the OS/390 shell. Following is a copy of the sample procedure:

```
//RSVPD    PROC
//*
//* SecureWay Communications Server IP
//* SMP/E distribution name: EZARSVPP
//*
//* 5647-A01 (C) Copyright IBM Corp. 1999.
//* Licensed Materials - Property of IBM
//*
//RSVPD    EXEC PGM=RSVPD,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")
//'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
```

```
//*            etc/rsvpd25.conf'
//*
//* Provide environment variables to run with the desired stack and
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*   RSVPD_CONFIG_FILE=/etc/rsvpd2.conf
//*   RSVPD_LOG_FILE=/tmp/rsvpd2.log
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.RSVPD.ENV(RSVPD2),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/rsvpd2.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, RSVPD doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the RSVPD_LOG_FILE environment variable, and defaults to
//* /tmp/rsvpd.log.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Stopping RSVP

RSVP can be stopped using the cancel command (C RSVPD) or using the kill command in the OS/390 shell. The following kill command with the TERM signal will enable RSVP to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is the RSVP process ID. The RSVP process ID can be obtained using the following OS/390 UNIX command:

```
ps -A
```

It can also be obtained from the /tmp/rsvpd.pid.imagename file.

Security Information

Because the RSVP Agent can affect system operation significantly, security product (for example, RACF) authority is required to start the RSVP Agent.

The following security product profile name must be defined:

```
MVS.SERVMGR.RSVPD
```

Listed below are sample commands used to create this profile name and permit users to it:

```
RDEFINE OPERCMDS (MVS.SERVMGR.RSVPD) UACC(NONE)
PERMIT MVS.SERVMGR.RSVPD ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Part 3. Using Translation Tables

Chapter 36. Using Translation Tables 1133
SBCS Translation Table Hierarchy 1133
Customizing SBCS Translation Tables 1135
ASCII-to-EBCDIC Table 1135
EBCDIC-to-ASCII Table 1135
Syntax Rules for SBCS Translation Tables 1136
SBCS Country Translation Tables. 1136
ISO-8 and IBM PC Interpretations for ASCII and EBCDIC Code Points . . .1137
DBCS Translation Table Hierarchy 1138
Usage Notes for the TRANSLATE Option for the FTP Client 1140
Telnet 3270 DBCS Transform Mode Codefiles 1140
Customizing DBCS Translation Tables 1141
DBCS Country Translation Tables 1141
Syntax Rules for DBCS Translation Tables 1142
Converting Translation Tables to Binary 1144
CONVXLAT Examples 1145

Chapter 36. Using Translation Tables

TCP/IP for MVS uses translation tables to convert transmitted data from EBCDIC to ASCII. Because these tables do not always include all the desired characters, TCP/IP allows you to create and customize tables without having to recompile source code. Translation tables are stored in binary form on disk. TCP/IP provides standard tables that are used as the default if you do not customize your own.

Two types of translation tables are used by TCP/IP for MVS. SBCS translation tables are used for single-byte characters. DBCS translation tables are used for converting double-byte characters. DBCS translation tables are required for character sets such as Japanese Kanji, which contains too many characters to represent using single-byte codes. SBCS translation tables provide mappings for a maximum of 256 characters. DBCS translation tables can provide up to a theoretical maximum of 65 535 character mappings; however, DBCS character sets usually contain less than this number.

The following sections describe how to create and customize both SBCS and DBCS translation tables and explain how they are used by the programs in TCP/IP for MVS.

SBCS Translation Table Hierarchy

Different programs look for special translation tables to use. The program chooses one of the customized tables, as described in Table 49. The program first searches for a customized table that you have built. If the program fails to find one of the customized tables, it uses the default table supplied in *hlq*.STANDARD.TCPXLBIN. Table 49 provides the customized translation tables and default table names for the programs.

Note: FTP server and FTP client will optionally use iconv instead of the external tables for single-byte conversion. The use of iconv is specified in FTP.DATA or by SITE/LOCSITE commands.

Table 49. SBCS Translation Table Hierarchy

Program	Customized Translation Tables	Default Translation Table
FTP Client	1. FTP.DATA keywords CTRLCONN and SBDATACONN 2. *user_id.FTP*.TCPXLBIN 3. *hlq.FTP*.TCPXLBIN 4. *user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
FTP Client (TRANSLATE) [4]	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN	None. Program Halts
FTP Server	1. DD:SYSFTSX in FTP start procedure 2. FTP.DATA keywords CTRLCONN (or CCXLATE) and SBDATACONN (or XLATE) 3. *jobname.SRVRFTP*.TCPXLBIN 4. *hlq.SRVRFTP*.TCPXLBIN 5. *jobname*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN

Table 49. SBCS Translation Table Hierarchy (continued)

Program	Customized Translation Tables	Default Translation Table
LPR Client	1. *user_id*.LPR.TCPXLBIN 2. *hlq*.LPR.TCPXLBIN 3. *user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPR Client (TRANSLATE)	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN 3. *user_id*.LPR.TCPXLBIN 4. *hlq*.LPR.TCPXLBIN 5. *user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPD Server	*user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPD Server (TRANSLATE)	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN	None. Program Halts
PORTMAP	1. *user_id*.STANDARD.TCPXLBIN 2. *jobname*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
REXEC	*user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
SMTP	1. *jobname.SMTP*.TCPXLBIN 2. *hlq.SMTP*.TCPXLBIN 3. *jobname*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
Telnet Client	1. *user_id*.TELNET.TCPXLBIN 2. *hlq*.TELNET.TCPXLBIN 3. *user_id*.STANDARD.TCPXLBIN	*hlq*.TELNET.TCPXLBIN
Telnet Client (TRANSLATE)	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN	None. Program Halts

Note:

- *jobname* is the name specified either on the PROC or JOB statement.
- *user_id* is the ID of the user who issued the command.
- *data_set* is the name entered on the TRANSLATE parameter for the program. See *OS/390 SecureWay Communications Server: IP User's Guide*for information on specifying the TRANSLATE parameter for the required program.

The Telnet client requires translation tables that are different from the default table *hlq*.STANDARD.TCPXLBIN. Customized translation tables for Telnet clients are provided in the install libraries as *hlq*.TELNET.TCPXLBIN and *hlq*.TELNETSE.TCPXLBIN. If these data sets are not found, however, then the Telnet client will use the default table.

The Telnet server (for Linemode) uses *iconv* services with the CODEPAGE statement in the TELNETPARMS block to specify country translation tables. TCPXLBIN translation tables are not used. If CODEPAGE is in error or not specified, refer to the CODEPAGE statement for default values used. If custom codepages are required, refer to the information about internationalization in the *IBM C/C++ for MVS/ESA Programming Guide* for details about how to create your own conversions.

4. Do not use the TRANSLATE option for the FTP client if the SBCS table you need for data transfer does not support standard encodings for the portable character set. Such a translation table can adversely affect the EBCDIC to ASCII conversion of commands sent over the control connection.

Customizing SBCS Translation Tables

All SBCS translation table members 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 A7, find row A0 (**3**) and column 07 (**4**) in "ASCII-to-EBCDIC Table". 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 X'7D' in EBCDIC.

To customize the translation table, alter the translate value where the row and column intersect to the new value.

You can edit and modify translation table members in the *hlq*.SEZATCPX data set.

ASCII-to-EBCDIC Table

```
;
; ASCII-to-EBCDIC table
;                      4                    2
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;                                              1
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 10 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; 20 ;
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; 30 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; 40 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; 50 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; 60 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; 70 ;
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 80 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 90 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; A0 ;  <- 3
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; B0 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; C0 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; D0 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; E0 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; F0 ;
```

EBCDIC-to-ASCII Table

```
;
; 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 0A 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.
- Information to the right of a semicolon (**;**) is a comment.

SBCS Country Translation Tables

Rather than customize the table in *hlq*.SEZATCPX(STANDARD), you can use the following translation table members, which are included with the code.

These translation table members are in the same format as that used in *hlq*.SEZATCPX(STANDARD). To use these table members, you must convert them to binary format using CONVXLAT and store the resulting binary tables in an appropriate data set within the SBCS translation table hierarchy. (See “SBCS Translation Table Hierarchy” on page 1133.) For more information about using the TSO CONVXLAT command, see “Converting Translation Tables to Binary” on page 1144.

The editable translation tables used by the Telnet client application are members of the *hlq*.SEZATELX data set and are derived from the identified code pages. The editable tables used by other applications, such as FTP, are members of the *hlq*.SEZATCPX data set.

The following members can be used by both Telnet client and non-Telnet SBCS applications such as FTP, SMTP, and so on. They will be found in both *hlq*.SEZATELX and *hlq*.SEZATCPX.

Note: Identically-named members in the two data sets are **not** the same.

Table 50. Translation Table Members for Telnet Client and Non-Telnet SBCS Applications

Member Name	Description	Code Page
AUSGER *	Austrian-German code page	850<->273
BELGIAN *	Belgian code page	850<->500
CANADIAN *	Canadian code page	850<->037
CUSTOM *	Code page	819<->1047
DANNOR *	Danish-Norwegian code page	850<->277
DUTCH *	Dutch code page	850<->037
FINSWED *	Finnish-Swedish code page	850<->278
FRENCH *	French code page	850<->297
ITALIAN *	Italian code page	850<->280
JAPANESE *	Japanese code page	850<->281
JPNALPHA	Japanese Code code page	1041<->1027
JPNKANA	Japanese Code code page	1041<->0290
KOR0891	Korean Code code page	0891<->0833
KOR1088	Korean Code code page	1088<->0833
PORTUGUE *	Portuguese code page	850<->037

Table 50. Translation Table Members for Telnet Client and Non-Telnet SBCS Applications (continued)

Member Name	Description	Code Page
PRC1115	P.R.China code page	1115<->0836
SPANISH *	Spanish code page	850<->284
SWISFREN *	Swiss-French code page	850<->500
SWISGERM *	Swiss-German code page	850<->500
TAI0904	Taiwan code page	0904<->0037
TAI1114	Taiwan code page	1114<->0037
UK *	United Kingdom code page	850<->285
US *	United States code page	850<->037

Footnote *: Refer to "ISO-8 and IBM PC Interpretations for ASCII and EBCDIC Code Points".

The following SBCS translation table members are only used by Telnet 3270 DBCS Transform support. These will be found only in *hlq*.SEZATELX.

Table 51. SBCS Translation Table Members for Telnet 3270 DBCS Transform Support

Member Name	Description	Code Page
A8E	Japanese 8-bit English	0819<->1027
A8K	Japanese 8-bit Katakana	0819<->0290
J8E	Japanese JIS 8-bit English	Unassigned
J8K	Japanese JIS 8-bit Katakana	Unassigend
SJDCE	Japanese DEC English	Unassigned
SJDCK	Japanese DEC Katakana	Unassigned
SJECE	Japanese Extended Unix English JIS	X0201<->1027
SJECK	Japanese Extended Unix Katakana JIS	X0201<->0290
KOR0891	Korean code page	0891<->0833
KOR1088	Korean code page	1088<->0833
PRC1115	P.R. China code page	1115<->0836
TAI0904	Taiwan code page	0904<->0037
TAI1114	Taiwan code page	1114<->0037

ISO-8 and IBM PC Interpretations for ASCII and EBCDIC Code Points

The tables in the *hlq*.SEZATCPX data set use the ISO-8 interpretations for certain ASCII code points. These code points are mapped to EBCDIC code points, as shown in the following table.

Table 52. ISO-8 Interpretations for Certain ASCII and EBCDIC Code Points

ASCII Code Point	EBCDIC Code Point	ISO-8 Interpretation
X'1A'	X'3F'	SUB (substitution character)
X'1C'	X'1C'	IFS (interchange file separator)
X'7F'	X'07'	DEL (delete character)

If you want to use IBM PC interpretations for these code points, you can modify your table, as shown:

Table 53. IBM PC Interpretations for Certain ASCII and EBCDIC Code Points

ASCII Code Point	EBCDIC Code Point	IBM PC Interpretation
X'1A'	X'1C'	IFS (interchange file separator)
X'1C'	X'07'	DEL (delete character)
X'7F'	X'3F'	SUB (substitution character)

DBCS Translation Table Hierarchy

Table 54 describes the search order used by certain programs when they are configured to load one or more DBCS translation tables. See the corresponding section on configuring the program, for information on which translation tables will be required.

If the customized DBCS translation tables are not found, then the default table data sets provided with the install libraries are used. If the default tables cannot be read, then error messages are issued, and the required DBCS conversion will be unavailable for the program.

Table 54. DBCS Translation Table Hierarchy

Program	Option	Customized Translation Tables	Default Translation Table
FTP Client	Hangeul	1. *user_id*.FTP.TCPHGBIN 2. *hlq*.FTP.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP Client	Kanji	1. *user_id*.FTP.TCPKJBIN 2. *hlq*.FTP.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP Client	SChinese	1. *user_id*.FTP.TCPSCBIN 2. *hlq*.FTP.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
FTP Client	TChinese	1. *user_id*.FTP.TCPCHBIN 2. *hlq*.FTP.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
FTP Client	Hangeul and TRANSLATE *	1. *user_id.data_set*.TCPHGBIN 2. *hlq.data_set*.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP Client	Kanji and TRANSLATE *	1. *user_id.data_set*.TCPKJBIN 2. *hlq.data_set*.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP Client	SChinese and TRANSLATE *	1. *user_id.data_set*.TCPSCBIN 2. *hlq.data_set*.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN

Table 54. DBCS Translation Table Hierarchy (continued)

Program	Option	Customized Translation Tables	Default Translation Table
FTP Client	TChinese and TRANSLATE *	1. *user_id.data_set*.TCPCHBIN 2. *hlq.data_set*.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
FTP Server	Hangeul	1. *jobname*.SRVRFTP.TCPHGBIN 2. *hlq*.SRVRFTP.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP Server	Kanji	1. *jobname*.SRVRFTP.TCPKJBIN 2. *hlq*.SRVRFTP.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP Server	SChinese	1. *jobname*.SRVRFTP.TCPSCBIN 2. *hlq*.SRVRFTP.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
FTP Server	TChinese	1. *jobname*.SRVRFTP.TCPCHBIN 2. *hlq*.SRVRFTP.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
LPR Client	Hangeul	1. *user_id*.LPR.TCPHGBIN 2. *hlq*.LPR.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
LPR Client	Kanji	1. *user_id*.LPR.TCPKJBIN 2. *hlq*.LPR.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
LPR Client	SChinese	1. *user_id*.LPR.TCPSCBIN 2. *hlq*.LPR.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
LPR Client	TChinese	1. *user_id*.LPR.TCPCHBIN 2. *hlq*.LPR.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
LPD Server	Hangeul	1. *jobname*.LPD.TCPHGBIN 2. *hlq*.LPD.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
LPD Server	Kanji	1. *jobname*.LPD.TCPKJBIN 2. *hlq*.LPD.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
LPD Server	SChinese	1. *jobname*.LPD.TCPSCBIN 2. *hlq*.LPD.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
LPD Server	TChinese	1. *jobname*.LPD.TCPCHBIN 2. *hlq*.LPD.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN

Table 54. DBCS Translation Table Hierarchy (continued)

Program	Option	Customized Translation Tables	Default Translation Table
SMTP Server	Hangeul	1. *jobname*.SMTP.TCPHGBIN 2. *hlq*.SMTP.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
SMTP Server	Kanji	1. *jobname*.SMTP.TCPKJBIN 2. *hlq*.SMTP.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
SMTP Server	SChinese	1. *jobname*.SMTP.TCPSCBIN 2. *hlq*.SMTP.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
SMTP Server	TChinese	1. *jobname*.SMTP.TCPCHBIN 2. *hlq*.SMTP.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN

Footnote *: Refer to "Usage Notes for the TRANSLATE Option for the FTP Client"

Note:

- *jobname* is the name specified either on the PROC or JOB statement.
- *user_id* is the ID of the user who issued the command.
- *data_set* is the name entered on the TRANSLATE parameter for the program. See the *OS/390 SecureWay Communications Server: IP User's Guide*for information on specifying the TRANSLATE parameter for the required program.

Usage Notes for the TRANSLATE Option for the FTP Client

1. To use the TRANSLATE option to load and use a customized DBCS translation table for the FTP client, an SBCS table data set must also exist for the *data_set_name* chosen with the TRANSLATE option.

   If the SBCS table data set does not exist, the FTP request will fail even if a valid DBCS table data set using that name exists.

2. ***CAUTION***: Do not use the TRANSLATE option for the FTP client if the SBCS table you need for data transfer does not support standard encodings for the portable character set. Such a translation table can adversely affect the EBCDIC to ASCII conversion of commands sent over the control connection.

   For information on using FTP.DATA to specify different SBCS tables for control and data connections, refer to *OS/390 SecureWay Communications Server: IP User's Guide*.

   If you require a local DBCS table, you can name it *userid*.FTP.TCPdbBIN and it will be found in the client's search order.

Telnet 3270 DBCS Transform Mode Codefiles

The binary translation table codefiles used by Telnet 3270 DBCS transform mode do not use a search order hierarchy. The codefile members must reside in a data set pointed to by the TNDBCSXL DD statement in the TCPIPROC cataloged procedure. If the DD statement is not specified, or the codefiles are not present, 3270 DBCS transform mode will be disabled.

Customizing DBCS Translation Tables

You can find the DBCS translation tables in the installation libraries in both editable source and binary form. The Kanji, Hanguel, Traditional Chinese and Simplified Chinese DBCS editable source members reside in the *hlq*.SEZADBCX data set. The standard binary members reside in the *hlq*.STANDARD.TCPKJBIN for Kanji, the *hlq*.STANDARD.TCPHGBIN for Hangeul, the *hlq*.STANDARD.TCPCHBIN for Traditional Chinese, and the *hlq*.STANDARD.TCPSCBIN for Simplified Chinese. These data sets contain binary tables that are used by the FTP server, SMTP server, FTP client, LPR client, and LPD server programs. The binary codefiles used by Telnet 3270 DBCS transform mode reside in the *hlq*.SEZAXLD2 data set. The binary tables and codefiles may be created from the same editable source, using the CONVXLAT program.

Steps to customize a DBCS translation table:

1. Make a copy of the editable source data set.
2. Modify the editable source as required.
3. Run the CONVXLAT program with the modified editable source as input.
4. Install the resulting customized binary table or codefiles in the DBCS translation table hierarchy for the required program.

The following sections show examples of the standard editable source for the Kanji, Hangeul, Simplified Chinese, and Traditional Chinese DBCS translation tables.

The editable source data sets contain 2 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 can be changed by simply overwriting the existing hexadecimal code. Code-points that are not defined in the target code page and are within the valid range for the code page are mapped to the default substitution character in the target code page. The default substitution characters are shown as (sub: xxxx) in the source tables on the following pages. For example, in “Editable Japanese KANJI DBCS Translation Tables” on page 1142, the default substitution character for SJISETA is FCFC.

The editable source format specifies EBCDIC-to-ASCII and ASCII-to-EBCDIC mappings 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 default substitution character.

DBCS Country Translation Tables

The following translation table source members are in the *hlq*.SEZADBCX data set. They are used by all applications that support DBCS. If you modify these table members, you must convert them to binary format using CONVXLAT and store the modified binary data set in an appropriate data set within the DBCS translation table hierarchy. See “Converting Translation Tables to Binary” on page 1144 for more information about using the TSO CONVXLAT command.

Table 55. Translation Table Members for DBCS Applications

Member Name	Description	Code Page
EZACHLAT (Taiwan DBCS)	TChinese Big5	0927<->0835 0947<->0835
EZAHGLAT (Korea DBCS)	Hangeul KSC5601	0926<->0834 0951<->0834
EZAKJLAT (Japan DBCS)	PC to host code page	PC<->0300
EZASCLAT(P.R.China DBCS)	Schinese	1380<->0837

Syntax Rules for DBCS Translation Tables

Comments can be included in the editable data set, either on a separate line or at the end of a line. Comments must start with a semicolon (";").

Code-point mappings in the data set are position dependent. The first non-comment line for the DBCS tables in the data set 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 area. 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 (that is, the "code index"), must be in ascending numeric order. Any gaps in the ascending order will be filled with the default substitution character in the binary table created by CONVXLAT.

Editable Japanese KANJI DBCS Translation Tables:

```
 ; ; EZAKJLAT TCPXLATE - Editable Japanese translate tables.
;
; ETA = Ebcdic to Ascii Conversion (Host Code Page ID 300 to PC)
; ATE = Ascii to Ebcdic Conversion (PC to Host Code Page ID 300)
;
; Use CONVXLAT to generate STANDARD TCPKJBIN (or TNDBCSTM codefiles)
; from this source file.
;
;DBCS Area - SJISETA,SJISATE not used for TNDBCSTM codefile generation.
;           - JDECETA,JDECATE not used for STANDARD TCPKJBIN generation.

;
;SJISETA    SJISATE     JIS78ETA   JIS78ATE    JIS83ETA   JIS83ATE    JEUCETA    JEUCATE     JDECETA    JDECATE
;
;sub: FCFC  sub: FEFE   sub: 747E  sub: FEFE   sub: 747E  sub: FEFE   sub: F4FE  sub: FEFE   sub: F4FE  sub: FEFE
;
 4040 8140  8140 4040   4040 2121  2121 4040   4040 2121  2121 4040   4040 A1A1  A1A1 4040   4040 A1A1  A1A1 4040
 4041 FCFC  8141 4344   4041 747E  2122 4344   4041 747E  2122 4344   4041 F4FE  A1A2 4344   4041 F4FE  A1A2 4344
 4042 FCFC  8142 4341   4042 747E  2123 4341   4042 747E  2123 4341   4042 F4FE  A1A3 4341   4042 F4FE  A1A3 4341
 4043 FCFC  8143 426B   4043 747E  2124 426B   4043 747E  2124 426B   4043 F4FE  A1A4 426B   4043 F4FE  A1A4 426B
 4044 FCFC  8144 424B   4044 747E  2125 424B   4044 747E  2125 424B   4044 F4FE  A1A5 424B   4044 F4FE  A1A5 424B
 4045 FCFC  8145 4345   4045 747E  2126 4345   4045 747E  2126 4345   4045 F4FE  A1A6 4345   4045 F4FE  A1A6 4345
 4046 FCFC  8146 427A   4046 747E  2127 427A   4046 747E  2127 427A   4046 F4FE  A1A7 427A   4046 F4FE  A1A7 427A
 4047 FCFC  8147 425E   4047 747E  2128 425E   4047 747E  2128 425E   4047 F4FE  A1A8 425E   4047 F4FE  A1A8 425E
 4048 FCFC  8148 426F   4048 747E  2129 426F   4048 747E  2129 426F   4048 F4FE  A1A9 426F   4048 F4FE  A1A9 426F
 4049 FCFC  8149 425A   4049 747E  212A 425A   4049 747E  212A 425A   4049 F4FE  A1AA 425A   4049 F4FE  A1AA 425A
 .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....
 .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....
 .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....   .... ....  .... ....
```

Editable Hangeul DBCS Translation Tables:

```
;
; EZAHGLAT TCPXLATE - Editable Korean DBCS translation tables.
;
```

```
; ETA = Ebcdic to Ascii Conversion (Host Code Page ID 834 to PC)
; ATE = Ascii to Ebcdic Conversion (PC to Host Code Page ID 834)
;
; Use CONVXLAT to generate STANDARD TCPHGBIN (or TNDBCSTM codefiles)
; from this source file.
;
;
; Code Page ID - 951         Code Page ID - 926
; KSCETA      KSCATE         HANETA      HANATE
;(sub: AFFE) (sub: FEFE)    (sub: BFFC) (sub: FEFE)
;
  4040 A1A1   8FA1 D541      4040 8140   8140 4040
  4041 AFFE   8FA2 D542      4041 BFFC   8141 4141
  4042 AFFE   8FA3 D543      4042 BFFC   8142 4142
  4043 AFFE   8FA4 D544      4043 BFFC   8143 4143
  4044 AFFE   8FA5 D545      4044 BFFC   8144 4144
  4045 AFFE   8FA6 D546      4045 BFFC   8145 4145
  4046 AFFE   8FA7 D547      4046 BFFC   8146 4146
  4047 AFFE   8FA8 D548      4047 BFFC   8147 4147
  4048 AFFE   8FA9 D549      4048 BFFC   8148 4148
  4049 AFFE   8FAA D54A      4049 BFFC   8149 4149
  .... ....   .... ....      .... ....   .... ....
  .... ....   .... ....      .... ....   .... ....
  .... ....   .... ....      .... ....   .... ....
```

Editable Traditional Chinese DBCS Translation Tables:

```
;
; EZACHLAT TCPXLATE - Editable Traditional Chinese DBCS translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host Code Page ID 835 to PC)
; ATE = Ascii to Ebcdic Conversion (PC to Host Code Page ID 835)
;
; Use CONVXLAT to generate STANDARD TCPCHBIN (or TNDBCSTM codefiles)
; from this source file.
;
;
; Code Page ID - 927         Code Page ID - 947
; TCHETA      TCHATE         BIG5ETA     BIG5ATE
;(sub: FCFC) (sub: FEFE)    (sub: C8FE) (sub: FEFE)
;
  4040 8140   8140 4040      4040 A140   8140 D649
  4041 FCFC   8141 4344      4041 C8FE   8141 D64A
  4042 FCFC   8142 4341      4042 C8FE   8142 D64B
  4043 FCFC   8143 426B      4043 C8FE   8143 D64C
  4044 FCFC   8144 424B      4044 C8FE   8144 D64D
  4045 FCFC   8145 4345      4045 C8FE   8145 D64E
  4046 FCFC   8146 427A      4046 C8FE   8146 D64F
  4047 FCFC   8147 425E      4047 C8FE   8147 D650
  4048 FCFC   8148 426F      4048 C8FE   8148 D651
  4049 FCFC   8149 425A      4049 C8FE   8149 D652
  .... ....   .... ....      .... ....   .... ....
  .... ....   .... ....      .... ....   .... ....
  .... ....   .... ....      .... ....   .... ....
```

Editable Simplified Chinese DBCS Translation Tables:

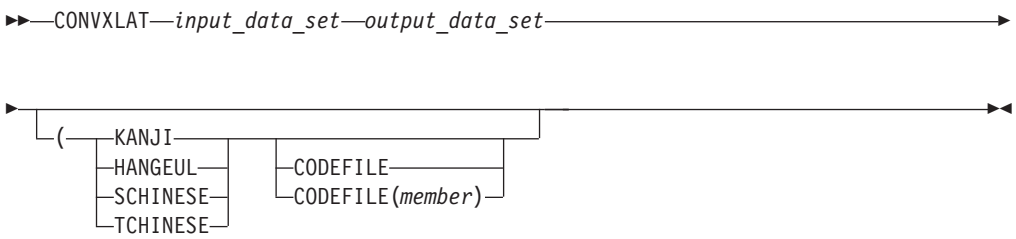
```
;
; EZASCLAT TCPXLATE - Editable Simplified Chinese DBCS translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host Code Page ID 837 to PC)
; ATE = Ascii to Ebcdic Conversion (PC to Host Code Page ID 837)
;
; Use CONVXLAT to generate STANDARD TCPSCBIN (or TNDBCSTM codefiles)
; from this source file.
;
;
; Code Page ID - 1380
```

```
; SCHETA      SCHATE
;(sub: FEFE) (sub: FEFE)
;
  4040 A1A1   8CA1 FEFE
  4041 FEFE   8CA2 FEFE
  4042 FEFE   8CA3 FEFE
  4043 FEFE   8CA4 FEFE
  4044 FEFE   8CA5 FEFE
  4045 FEFE   8CA6 FEFE
  4046 FEFE   8CA7 FEFE
  4047 FEFE   8CA8 FEFE
  4048 FEFE   8CA9 FEFE
  4049 FEFE   8CAA FEFE
  .... ....   .... ....
  .... ....   .... ....
  .... ....   .... ....
```

Converting Translation Tables to Binary

The TSO CONVXLAT command converts a table from editable text to binary. CONVXLAT can be used to convert both SBCS and DBCS table source data sets.

The syntax of the CONVXLAT command is:



The parameters of the CONVXLAT command are:

input_data_set
: Specifies the source data set name to be converted. The data set name must be enclosed in quotes if fully qualified, otherwise the TSO user ID is appended as a prefix.

output_data_set
: Specifies the destination data set name created by the conversion. The data set name must be enclosed in quotes if fully qualified, otherwise the TSO user ID is appended as a prefix.

  If CODEFILE is also specified, then *output_data_set* must specify a previously allocated partitioned data set. Multiple codefile members will be placed in the partitioned data set. The data set should be allocated using the following parameters:

```
Organization:          PO
Record format:         VB
Record length:         5124
Block size:            8800
1st extent blocks:     156
Secondary blocks:      10
```

KANJI Specifies that the tables being converted are the Japanese DBCS translation tables.

HANGEUL
Specifies that the tables being converted are the Korean Standard DBCS translation tables.

SCHINESE
Specifies that the table being converted is the Simplified Chinese DBCS translation table.

TCHINESE
Specifies that the tables being converted is the Traditional Chinese DBCS translation table.

CODEFILE
Specifies that the selected table is converted to multiple codefiles for use in Telnet 3270 DBCS transform mode. The selected table must be DBCS translation table.

CODEFILE(*member***)**
Specifies that the selected SBCS table is converted to two codefiles: ASCII_To_EBCDIC and EBCDIC_To_ASCII. The member names in the output PDS are *member*ATE and *member*ETA. The possible member names are:

J8E JIS 8 Bit English

J8K JIS 8 Bit Katakana

A8E 8 Bit English

A8K 8 Bit Katakana

SJDCE
DEC English SBCS

SJDCK
DEC Katakana SBCS

SJECE
Japanese EUC English SBCS

SJECK
Japanese EUC Katakana SBCS

SKSH Korean KSC 5601 SBCS

SHAN Hangeul SBCS

STCH Traditional Chinese SBCS

SBG5 Big-5 SBCS

SSCH Simplified Chinese SBCS

If no optional parameters are specified, then the input data set is assumed to contain an SBCS translation table.

CONVXLAT Examples

Running CONVXLAT in BATCH: The following is an example of running CONVXLAT in batch:

```
//S00100    EXEC PGM=CONVXLAT,
//          PARM='''TCPIP.V3R1.SEZATCPX(STANDARD)'' STANDARD.TCPXLBIN'
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY,BLKSIZE=80
```

SBCS Binary Table: The following example shows the creation of a SBCS binary table from user-provided editable text.

```
convxlat sbcs.source  standard.tcpxlbin
READY
```

French Telnet Client SBCS: The following example shows the creation of a French Telnet Client SBCS binary table for userid user30 from the product provided editable text.

```
convxlat 'tcpip.v3r2.sezatelx(french)' 'user30.telnet.tcpxlbin'
READY
```

Korean KSC5601 SBCS and DBCS: The following example shows the creation of a Korean KSC5601 SBCS and DBCS binary table from the product-provided editable text. These tables can be used by FTP, LPR, LPD and SMTP.

```
convxlat 'tcpip.v3r2.sezatcpx(kor1088)' 'tcpip.v3r2.standard.tcpxlbin'
READY
convxlat 'tcpip.v3r2.sezadbcx(ezahglat)' 'tcpip.v3r2.standard.tcphgbin'
(hangeul
READY
```

Big-5 and Traditional Chinese: The following example shows the creation of Big-5 and Traditional Chinese SBCS and DBCS codefiles for use by the Telnet 3270 DBCS Transform facility.

```
convxlat 'tcpip.v3r2.sezatelx(TAI1114)'  'tcpip.v3r2.sezaxld2' (codefile(sbg5)
READY
convxlat 'tcpip.v3r2.sezatelx(TAI0904)'  'tcpip.v3r2.sezaxld2' (codefile(stch)
READY
convxlat 'tcpip.v3r2.sezadbcx(ezachlat)'   'tcpip.v3r2.sezaxld2' (tchinese
codefile
EZA0652I Current code set is "TCHETA"
EZA0652I Current code set is "TCHATE"
EZA0652I Current code set is "BG5ETA"
EZA0652I Current code set is "BG5ATE"
READY
```

Japanese SBCS (CP 1041) and DBCS: The following example shows the creation of a Japanese SBCS (CP 1041) and DBCS binary table from the product provided editable text. These tables can be used by FTP, LPR, LPD and SMTP.

```
convxlat 'tcpip.v3r2.sezatcpx(JPNKANA)'  'tcpip.v3r2.standard.tcpxlbin'
READY
convxlat 'tcpip.v3r2.sezadbcx(ezakjlat)'  'tcpip.v3r2.standard.tcpkjbin'
(kanji
READY
```

Japanese SBCS and DBCS Codefiles: The following example shows the creation of Japanese SBCS and DBCS codefiles for use by the Telnet 3270 DBCS Transform facility.

```
convxlat 'tcpip.v3r2.sezatelx(J8E)'  'tcpip.v3r2.sezaxld2' (codefile(j8e)
READY
convxlat 'tcpip.v3r2.sezatelx(J8K)'  'tcpip.v3r2.sezaxld2' (codefile(j8k)
READY
convxlat 'tcpip.v3r2.sezatelx(A8E)'  'tcpip.v3r2.sezaxld2' (codefile(a8e)
READY
convxlat 'tcpip.v3r2.sezatelx(A8K)'  'tcpip.v3r2.sezaxld2' (codefile(a8k)
READY
convxlat 'tcpip.v3r2.sezatelx(SJECE)'   'tcpip.v3r2.sezaxld2' (codefile(sjece)
READY
convxlat 'tcpip.v3r2.sezatelx(SJECK)'   'tcpip.v3r2.sezaxld2' (codefile(sjeck)
READY
convxlat 'tcpip.v3r2.sezatelx(SJDCE)'   'tcpip.v3r2.sezaxld2' (codefile(sjdce)
READY
convxlat 'tcpip.v3r2.sezatelx(SJDCK)'   'tcpip.v3r2.sezaxld2' (codefile(sjdck)
READY
convxlat 'tcpip.v3r2.sezadbcx(ezakjlat)'   'tcpip.v3r2.sezaxld2' (kanji
codefile
EZA0652I Current code set is "JIS78ETA"
EZA0652I Current code set is "JIS78ATE"
EZA0652I Current code set is "JIS83ETA"
EZA0652I Current code set is "JIS83ATE"
EZA0652I Current code set is "JEUCETA"
EZA0652I Current code set is "JEUCATE"
EZA0652I Current code set is "JDECETA"
EZA0652I Current code set is "JDECATE"
READY
```

Part 4. Appendixes

Appendix A. HFS File Requirements

The terms *data sets* and *files* are comparable. If you are familiar with MVS, you probably use the term *data set* to describe a unit of data storage. If you are familiar with AIX or UNIX, you probably use the term *file* to describe a named set of records stored or processed as a unit. In the CS for OS/390 environment, there are several product parts that reside in an HFS. If you are unfamiliar with hierarchical file systems, read the following section.

Note: The CS for OS/390 HFS must be configured and mounted for proper operation of the TCP/IP stack and applications. This is a requirement even if you do not plan on directly exploiting applications in the OS/390 UNIX environment.

OS/390 UNIX Hierarchical File System Concepts

The Hierarchical File System lets you to set up a file hierarchy that consists of:

- HFS **files**, which contain data or programs. A file containing a load module, shell script or REXX program is called an *executable file*. Files are kept in directories.
- **Directories** that contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with root directory at the top and the branches at the bottom. The **root** is the first directory for the file system at the peak of the tree and is designated by a slash (/).
- Additional local or remote **file systems**, that are mounted on directories of the root file system or of additional file systems.

To the OS/390 system, the file hierarchy is a collection of hierarchical file system (HFS) data sets. Each HFS data set is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be logically mounted on a directory within the root file system or on a directory within any mounted file system.

Except for the direction of the slashes, the hierarchical file system is similar to a Disk Operating System (DOS) or an OS/2* file system.

Each *mountable* file system resides in a hierarchical file system (HFS) data set on direct access storage. DFSMS/MVS manages the HFS data sets and the physical files.

The Root File System

The root system is the starting point for the overall HFS file structure. It contains the root directory and any related HFS files or subdirectories. You create the root file system when you install OS/390. The following section describes the specific locations where the parts of CS for OS/390 reside.

Mounting the CS for OS/390 HFS

The CS for OS/390 HFS must be mounted prior to starting up the TCP/IP stack. This can be accomplished by updating the BPXPRMxx member in effect for the OS/390 UNIX environment. For more information, refer to the *Program Directory for OS/390 Version 2 Release 5 for CBPDO Installation and ServerPac Reference* or ServerPac *Installing Your Order*.

Location of CS for OS/390 Files in the HFS

Table 56 shows the location of the CS for OS/390 parts. Executables marked with the number ″1″ exist only in the HFS with the sticky bit off. All the remaining executables exist only in an MVS data set. The HFS contains a non-executable module with the sticky bit on. If the sticky bit is on, these modules are fetched from CS for OS/390 MVS library data sets (for example, TCPIP.SEZALINK).

Note: Using the command `chmod` to alter the sticky bit setting for an executable file name in the HFS not only alters the setting for the specific file, but all other executables as well. If the sticky bit is turned off, the executable will not run since the file in the HFS is only used to point to the member in the PDS.

Table 56. CS for OS/390 Parts

Item	Location	Data set
FTP client	/bin/ftp	SEZALINK
OS/390 UNIX Netstat client	/bin/onetstat	SEZALINK
OS/390 UNIX PING client	/bin/oping	SEZALINK
OS/390 UNIX REXEC client	/bin/orexec	SEZALINK
OS/390 UNIX SNMP client	/bin/osnmp	SEZALINK
OS/390 UNIX Traceroute	/bin/otracert	SEZALINK
OS/390 UNIX SNMP pwtokey command	/bin/pwtokey	SEAZLINK
OS/390 UNIX SNMP pwchange command	/bin/pwchange	SEAZLINK
OS/390 UNIX REXECD server	/usr/sbin/orexecd	SEZALINK
OS/390 UNIX OMPROUTE application	/usr/sbin/omproute	SEZALINK
OS/390 UNIX OROUTED server	/usr/sbin/orouted	SEZALINK
OS/390 UNIX RSHD server	/usr/sbin/orshd	SEZALINK
OS/390 UNIX SNMP server	/usr/sbin/osnmpd	SEZALINK
OS/390 UNIX TelnetD server	/usr/sbin/otelnetd	SEZALINK
OS/390 Service Policy Agent	/usr/sbin/pagent	SEZALINK
OS/390 RSVP Agent	/usr/sbin/rsvpd	SEZALINK
OS/390 SLA subagent	/usr/sbin/pagtsnmp	SEZALINK
Syslogd server	/usr/sbin/syslogd	SYS1.LINKLIB SEZALINK SEZALINK
DHCP Admin interface server	/usr/sbin/dadmin	SEZALINK
DHCP server	/usr/sbin/dhcpsd	SEZALINK
BINL server	/usr/sbin/binlsd	SEZALINK
Trivial FTP server	/usr/sbin/tftpd	SEZALINK
Timed server	/usr/sbin/timed	SEZALINK
OS/390 UNIX sendmail		
sendmail	/usr/sbin/sendmail (1)	n/a
mailstats	/usr/sbin/mailstats (1)	n/a
popper	/usr/sbin/popper	SEZALINK
Samples	/usr/lpp/tcpip/samples	n/a
OS/390 UNIX FTPD server		

Table 56. CS for OS/390 Parts (continued)

Item	Location	Data set
Listener	/usr/sbin/ftpd	SEZALINK
Server	/usr/sbin/ftpdns	SEZALINK
HSAS (High Speed Access Services)		
NETSTAT command	/bin/oenetstat (1)	n/a
PING command	/bin/oeping	SEZALINK
Config command	/usr/sbin/oeifconfig (1)	n/a
Route command	/usr/sbin/oeroute	SEZALINK
OS/390 UNIX ONC/RPC		
Portmapper command	/bin/oportmap	SEZALINK
ONC RPC protocol compiler	/bin/orpcgen	SEZALINK
RPCINFO command	/bin/orpcinfo	SEZALINK
Header files	/usr/include/rpc	n/a
Library archive	/usr/lib/librpclib.a	n/a
Sample code	/usr/lpp/tcpip/rpc/samples	n/a
BIND-based DNS		
nsupdate client	/bin/nsupdate	SEZALINK
onslookup client	/bin/onslookup (1)	n/a
named server	/usr/sbin/named	SEZALINK
namedxfr	/usr/sbin/namedxfr	SEZALINK
X Window Systems		
Header files	/usr/include	n/a
Library archives	/usr/lib	n/a
Uil Compiler	/bin/X11/uil (1)	n/a
Sample code	/usr/lpp/tcpip/X11R6/Xamples	n/a

Directories or Files

In addition to the files related to CS for OS/390 executables listed in Table 56 on page 1152, the following files are used if available:

/bin/ls This file is provided by OS/390 UNIX and is required by the FTP server.

/dev/null
This file is provided by OS/390 UNIX and is required by the FTP server.

/dev/ptypXXXX and dev/ttypXXXX
These special device files represent pseudoterminals (ptys); they are used by the OS/390 UNIX Telnet server and other programs.

/etc/banner
A file used by the OS/390 UNIX Telnet server. If -h is not specified in the /etc/inetd.conf, then an additional banner is expected to be printed to the client's screen. This banner should be stored here.

/etc/ftp.data
This is the FTP server configuration file.

/etc/gateways
The gateways file is used to configure the routing table on the host system running RouteD (ORouteD).

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file. Some of the applications started by inetd are OS/390 UNIX Telnet, rexecd, and rshd.

/etc/mibs.data
This file defines textual names for MIB variables. It is used by the osnmp command.

/etc/osnmpd.data
This file contains configuration information pertaining to osnmpd.

/etc/pagent.conf
Default Policy Agent configuration file.

/etc/rsvp.conf
This is the default RSVP Agent configuration file.

/etc/resolv.conf
The OS/390 UNIX applications reference this file (or equivalent file specified by the environment variable RESOLVER_CONFIG) to determine the data set prefix, message case setting, and the DBCS translate tables that are to be loaded.

/etc/services
The ports for each OS/390 UNIX application are defined here.

/etc/snmpd.conf
This file provides configuration information to the SNMP agent when SNMPv3 support is used.

/etc/snmptrap.dest
This file provides a list of entities that should receive SNMP traps. It is used by osnmpd.

/etc/snmpv2.conf
This file provides configuration information for use by the osnmp command.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file. For example, the daemon.* entry in this file determines where FTP error messages and trace entries are recorded.

/usr/lib/nls/msg/C/cfmsg.cat
This is the default message catalog used by the configuration component. It is used for system operator commands. **If it is not found, TCP/IP initialization fails.**

/usr/lib/nls/msg/C/dadmsg.cat
This is the default message catalog used by the DHCP administration function. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/dhcpsd.cat
This is the default message catalog used by the DHCP Server. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/msrpcgen.cat

This is the default message catalog used by the orpcgen command. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/msrpclib.cat

This is the default message catalog used by the ONC RPC calls. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/syslogd.cat

This is the default message catalog used by the Syslog server. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/tftpdmsg.cat

This is the default message catalog used by the Trivial FTP server. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/timedmsg

This is the default message catalog used by the Time server. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/xm12.cat

This is the default message catalog used by the X11R6 Motif interface. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will

default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/Mrm12.cat

This is the default message catalog used by the X11R6 windowing system. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/NS.cat

This is the default message catalog used by the bind-based domain name server. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/NSLK.cat

This is the default message catalog used by the onslookup command. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/NSUPDATE.cat

This is the default message catalog used by the nsupdate command. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/RES.cat

This is the default message catalog used by the bind-based domain name server's resolver. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/Uil12.cat

This is the default message catalog used by the X11R6 windowing system. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the

LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/X11R6.cat

This is the default message catalog used by the X11R6 windowing system. It must be in the appropriate directory within the nlspath.

Where in the HFS the message catalog (msg.cat) is looked for depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/ftpdmsg.cat

This is the default FTP message catalog used by the FTP server. It must be in the appropriate directory within the nlspath.

Where the server looks for the message catalog (msg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/ftpdrply.cat

This is the default reply catalog used by the FTP server.

There can be up to 10 reply catalogs. See the *OS/390 SecureWay Communications Server: IP User's Guide*.

/usr/lib/nls/msg/C/netmsg.cat

This is the default message catalog for the onetstat command.

Where the command looks for the message catalog (netmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If netmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/omprdmsg

This is the default message catalog used by OMPROUTE.

Where the command looks for the message catalog (omprdmsg) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the pingmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/pingmsg.cat

This is the default message catalog used by the oping command.

Where the command looks for the message catalog (pingmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the pingmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/rexcmsg.cat

The message catalog associated with the OS/390 UNIX REXEC client is

stored here. If this file does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/rexdmsg.cat
This is the message catalog used by the OS/390 UNIX REXECD server.

Where the server looks for the message catalog (rexdmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the rexdmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/routed.cat
This is the default message catalog used by the OROUTED server.

Where the server looks for the message catalog (routed.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the routed.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/rshdmsg.cat
The message catalog associated with the OS/390 UNIX RSHD client is stored here. If this file does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/snmpclim.cat
This is the default message catalog for the osnmp command.

Where the command, osnmp, looks for the message catalog (snmpclim.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the snmpclim.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/snmpdmsg.cat
This is the default message catalog for the SNMP agent. This message catalog is also used for the pwtokey and pwchange commands.

Where the agent looks for the message catalog (snmpdmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the snmpdmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/subamsg.cat
This is the default message catalog for the SNMP subagent.

Where the subagent looks for the message catalog (subamsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the subamsg.cat does not exist, the

software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/pagtsmsg.cat
This is the default message catalog for the SLA subagent.

/usr/lib/nls/msg/C/tnmsgs.cat
The message catalog used by the OS/390 UNIX Telnet server is stored here.

Where the server looks for the message catalog (msg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cat elsewhere, you need to change the NLSPATH or the LANG environment variables. If the msg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/nls/msg/C/trtemsg.cat
This is the default message catalog for the otracert command.

Where the command looks for the message catalog (otracert.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If the trtemsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/lib/terminfo
This file is used by the OS/390 UNIX Telnet server. The terminal information that is verified during the tgetent call is stored here. This file requires permissions of 644. To expand it, issue the following command:

```
tic /usr/lib/terminfo/ibm.ti
```

Note: This should have been done during system setup. If this is not the first time that the tic command has been issued, you may have difficulty re-issuing this command. Make sure that you delete the symbolic links to /usr/lib/terminfo prior to re-issuing the command; then, reset the symbolic links.

/usr/lpp/tcpip/samples/db.data
Sample forward domain data file for BIND-Based DNS

/usr/lpp/tcpip/samples/dhcpsd
Sample DHCP server configuration file

/usr/lpp/tcpip/samples/dnswlm.readme
This 'read me' is for the BIND-Based Domain Name Server

/usr/lpp/tcpip/samples/dpi_mvs_sample.c
Sample DPI version 2.0 subagent code

/usr/lpp/tcpip/samples/dpiSimpl.mi2
SNMP v2 syntax (SMI) for the DPISimple-MIB objects supported by the sample DPI version 2.0 subagent

/usr/lpp/tcpip/samples/ibm3172.mib
SNMPv1 syntax (SMI) for the 3172 enterprise specific MIB objects

/usr/lpp/tcpip/samples/ibm3172.mi2
SNMPv2 syntax (SMI) for the 3172 enterprise specific MIB objects

/usr/lpp/tcpip/samples/iwmwdnsh.h
Sample wlm header file for WLM user applications

/usr/lpp/tcpip/samples/mibs.data
Sample /etc/mibs.data configuration file for use by the osnmp command

/usr/lpp/tcpip/samples/mvstcpip.caps
Formal SNMPv2 definition of the MIBs supported by the SNMP agent and subagent in CS for OS/390

/usr/lpp/tcpip/samples/mvstcpip.mib
SNMPv1 syntax (SMI) for the IBM MVS enterprise specific MIB objects

/usr/lpp/tcpip/samples/mvstcpip.mi2
SNMPv2 syntax (SMI) for the IBM MVS enterprise specific MIB objects

/usr/lpp/tcpip/samples/slapm.mib
SNMPv1 SMI for SLA subagent

/usr/lpp/tcpip/samples/slapm.mi2
SNMPv2 SMI for SLA subagent

/usr/lpp/tcpip/samples/slapm.txt
Text of the SLAPM-MIB draft RFC

/usr/lpp/tcpip/samples/named.boot
Sample boot file for BIND-Based DNS

/usr/lpp/tcpip/samples/named.lbk
Sample loopback file for BIND-Based DNS

/usr/lpp/tcpip/samples/named.rev
Sample reverse domain data file for BIND-Based DNS

/usr/lpp/tcpip/samples/osnmpd.data
Sample /etc/osnmpd.data configuration file for use by the SNMP agent (OSNMPD)

/usr/lpp/tcpip/samples/pagent.conf
Sample Policy Agent configuration file.

/usr/lpp/tcpip/samples/pagent.ldif
Sample policy definitions for Lightweight Directory Access Protocol (LDAP) server.

/usr/lpp/tcpip/samples/pagent_at.conf
Sample policy attributes defined for Lightweight Directory Access Protocol (LDAP) server.

/usr/lpp/tcpip/samples/pagent_oc.conf
Sample policy object class definitions defined for Lightweight Directory Access Protocol (LDAP) server.

/usr/lpp/tcpip/sample/rsvpd.conf
Sample RSVP Agent configuration file.

/usr/lpp/tcpip/samples/protocol
Sample /etc/protocol file

/usr/lpp/tcpip/samples/rfc1592b.mib
SNMPv2 syntax (SMI) for the expanded implementation of RFC 1592 supported by the SNMP agent in CS for OS/390

/usr/lpp/tcpip/samples/rfc1592b.mi2
SNMPv1 syntax (SMI) for the expanded implementation of RFC 1592 supported by the SNMP agent in CS for OS/390

/usr/lpp/tcpip/samples/saMIB.mib
SNMPv1 syntax (SMI) for the subagent MIB (saMIB) objects supported by the SNMP agent in CS for OS/390

/usr/lpp/tcpip/samples/saMIB.mi2
SNMPv2 syntax (SMI) for the subagent MIB (saMIB) objects supported by the SNMP agent in CS for OS/390

/usr/lpp/tcpip/samples/sendmail/README.m4
Contains all the latest information regarding this latest version of sendmail from the *www.sendmail.org* site.

/usr/lpp/tcpip/samples/sendmail/cf
Both site-dependent and site-independent descriptions of hosts. Files ending in **.mc** (″Master Configuration″) are the input descriptions. The output is in the corresponding **.cf** file.

/usr/lpp/tcpip/samples/sendmail/domain
Site-dependent subdomain descriptions. These are tied to the way your organization wants to do addressing. These descriptions are referenced using the DOMAIN *m4* macro in the **.mc** file.

/usr/lpp/tcpip/samples/sendmail/feature
Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE *m4* macro. An example feature is `use_cw_file`, which tells OS/390 UNIX sendmail to read an `/etc/sendmail.cw` file on startup to find the set of local names.

/usr/lpp/tcpip/samples/sendmail/hack
Local hacks, referenced using the HACK *m4* macro. Avoid these.

/usr/lpp/tcpip/samples/sendmail/m4
Site-independent *m4(1)* include files that have information common to all configuration files. Think of this as a ″#include″ directory.

/usr/lpp/tcpip/samples/sendmail/mailer
Definitions of mailers, referenced using the MAILER *m4* macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use ″MAILER(uucp)″.

/usr/lpp/tcpip/samples/sendmail/ostype
Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE *m4* macro. This directory contains only the os390 definition.

/usr/lpp/tcpip/samples/sendmail/sh
Shell files used by the *m4* build process.

/usr/lpp/tcpip/samples/sendmail/siteconfig
Local UUCP connectivity information. These normally contain lists of site information, such as SITE(contessa), SITE(hoptoad), SITE(nkainc), and SITE(well). These are referenced using the SITECONFIG macro. This directory has been supplanted by the mailer table feature. Any new configurations should use that feature to do UUCP (and other) routing.

/usr/lpp/tcpip/samples/sendmail/sendmail.ps
This is a PS file of the *Sendmail Installation and Operation Guide* provided by *www.sendmail.org* in this version of sendmail.

/usr/lpp/tcpip/samples/services
Sample /etc/services file

/usr/lpp/tcpip/samples/snmpd.conf
Sample /etc/snmpd.conf configuration file for use by the SNMP agent (OSNMPD)

/usr/lpp/tcpip/samples/snmpv2.conf
Sample /etc/snmpv2.conf configuration file for use by the osnmp command

/usr/lpp/tcpip/samples/syslog.conf
Sample /etc/syslog.conf file

/usr/lpp/tcpip/samples/tcpsym
Sample command file to re-establish CS for OS/390 symbolic links

/usr/lpp/tcpip/samples/wlmreg.c
Sample C interfaces for WLM registration and deregistration

/usr/man/C/cat1/onetstat.1
This file contains the manual (man) page for the onetstat command. It provides online help for the user.

/usr/man/C/cat1/oping.1
This file contains the manual (man) page for the oping command, oping. It provides online help for the user.

/usr/man/C/cat1/orexec.1
This file contains the manual (man) page for the orexec command. It provides online help for the user.

/usr/man/C/cat1/osnmp.1
: This file contains the manual (man) page for the osnmp command. It provides online help for the user.

/usr/man/C/cat1/otelnetd.1
: This file contains the associated manual (man) pages for the otelnetd server. It provides online help for the user.

/usr/man/C/cat1/otracert.1
: This file contains the manual (man) page for the otracert command. It provides online help for the user.

/usr/man/C/cat1/uil.1
: This file contains the manual (man) page for the user interface language compiler. It provides online help for the user.

Appendix B. SMF Records

This appendix describes the SMF records for the Telnet and FTP servers, API calls, and FTP and Telnet client calls. The EZASMF76 macro can be used to map the TCP/IP SMF records. EZASMF76 produces assembler level DSECTs for the Telnet (Server and Client), FTP (Server and Client), and API SMF records.

To create the Telnet SMF Record layout, code:

```
EZASMF76 TELNET=YES
```

To create the FTP SMF Record layout, code:

```
EZASMF76 FTP=YES
```

To create the API SMF Record layout, code:

```
EZASMF76 API=YES
```

Standard Subtype Record Numbers

TCP/IP logging of SMF records can be activated through the use of the SMFCONFIG and SMFPARMS statements in the TCP/IP profile. TCP/IP SMF records are written using record type 118 (X'76'). TCP/IP uses the standard subtype record numbers shown in this section.

Note: If you use the SMFPARMS statement, you can specify that records be written with non-standard subtype records. However, it is recommended that you use the standard subtype records shown in Table 57.

Table 57. Standard Subtype Record Numbers

Record number	Description
1	TCP API initialization
2	TCP API termination
3	FTP client
4	TN3270 client
5	TCP/IP statistics
6-19	Reserved
20	TN3270 server initialization
21	TN3270 server termination
22-29	Reserved
30	DNS statistics
31-69	Reserved
70	FTP server append subcommand
71	FTP server delete subcommand
72	FTP server logon failures
73	FTP server rename
74	FTP server retrieve
75	FTP server store
76–255	Reserved

Telnet Server SMF Record Layout

The SMF record written by the Telnet server has the following format:

Table 58. Telnet Server SMF Record Format

Byte	Subfield (Offset)	Description
0–23		Standard SMF header
	SMFxFLG (4)	A system indicator that is set to 66 (X'42')
	SMFxRTY (5)	A record type that is set to 118 (X'76') for all TCP/IP records
	SMFxSTY (22)	The record subtype obtained from the SMF keywords on the TCP/IP TELNETPARMS parameter. The subtype value is in the range X'0' –X'FF'.
24–27		LOGN or LOGF for session initiation/termination
28–35		8-character LU name
36–43		8-character application name
44–47		Internal logical device address (same for logon or logoff records)
48–51		Fullword remote IP address
52–55		Fullword local IP address
56–63		Started task qualifier name, for example, TCPIP
64–71		TCP/IP host name
72–73		Reserved
74–77		IN byte count
78–81		OUT byte count
82–85		Time specified in hundredths of a second on LOGF record
86–89		Julian date specified in packed decimal format on LOGF record
90–91		Remote port number
92–93		Local port number

FTP Server SMF Record Layout

The SMF record written by the FTP server has the following format:

Table 59. FTP Server SMF Record Format

Byte	Subfield (offset)	Description
0–23		Standard SMF header
	SMFxFLG (4)	A system indicator. If the first bit is ON, record subtypes are valid.
	SMFxRTY (5)	A record type that is set to 118 (X'76') for all TCP/IP records
	SMFxSTY (22)	The record subtype obtained from the SMF statements in the FTP.DATA data set. The subtype value is in the range X'0' – X'FF'.
24–27		4-byte FTP subcommand (for example, STOR, REN, DELE)
28–31		4-byte FTP file type (SEQ, JES, SQL)
32–35		Fullword client IP address
36–39		Fullword server IP address
40–47		Reserved
48–55		Local user ID

Table 59. FTP Server SMF Record Format (continued)

Byte	Subfield (offset)	Description
56		Data format (A—ASCII, E—EBCDIC, I—Image (binary), B—Double-byte, and U—UCS-2))
57		Mode (S—stream , B—block, C—compressed)
58		Structure (F—file)
59		Data set type (P—partitioned, blank—sequential, H—HFS)
60–63		Start time of transmission
64–67		End time of transmission
68–71		Byte count of transmission
72		FTP ID (S—server)
73–75		Last reply sent to this client (FTP server)
76–119		For LOGIN records, this is the user ID of the failed login attempt. Otherwise, this is the data set name or up to the first 44 bytes of the HFS file name.
120–127		Member name for PDS
128–135		Reserved for abnormal end information
136–179		Second data set name, if needed (for example, Rename). For HFS, up to the first 44 bytes of the HFS file name.
180–187		Second member name, if needed (for example, Rename)
188–195		Started task qualifier
196–203		TCP/IP host name
204–205		Remote port number
206–207		Local port number
208–209		Offset to the first HFS file name field
210–211		Offset to the second HFS file name field

Two variable-length fields at the end of the record contain HFS file names. The variable-length HFS name fields have the following format:

Byte	Description
0–1	Length of the HFS file name.
3–x	HFS file name.

The value in the length field is the length of the HFS file name only; it does not include the two bytes for the length field itself. The maximum size of this variable field is 1025 bytes; the maximum value in the length subfield is 1023.

SMF Record Layout for API Calls

The SMF record written by API calls for sockets has the following format:

Table 60. API Call SMF Record Format

Byte	Description
0–23	Standard SMF header with subtypes
24–27	Status of the connection, INIT, or TERM
28–31	Local IP address

Table 60. API Call SMF Record Format (continued)

Byte	Description
32–35	Foreign IP address
36–37	Local port number
38–39	Foreign port number
40–43	Bytes in. This is valid only for termination.
44–47	Bytes out. This is valid only for termination.
48–49	User area offset. Start of an area available for user exit storage.
50–51	User area length. Maximum length of user data, which is 52 bytes, for TCP/IP V3R2 for MVS. This size might increase between TCP/IP releases.
52–59	Job name: • For interactive TSO API usage: the user's TSO user ID • For batch-submitted jobs: the name of the JOB card • For started procedures: the name of the procedure.
60–67	Job identification (for HPNS applications only. (See note 1.) For other applications, this field is filled with blanks.) The JES job identifier.
68–71	Job start time (for HPNS applications only. (See note 1.) For other applications, this field is binary zeroes.) Time in hundredths of seconds that the job was started by JES.
72–75	Job date (for HPNS applications only. (See note 1.) For other applications, this field is binary zeroes.) Date job was started by JES. The date is in the form of `0CYYDDDF` where `C` is 0 for 19`yy` and 1 for 20`yy`, `DDD` is the day of the year (1-365), and `F` is the sign. For TSO/E, it is the logon enqueue date.
76–127	User area. Available for user exit usage. **Note:** The actual displacement of this area might change between TCP/IP releases. Use the values of the user area offset and the user area length fields to access this area.
Note 1: The HPNS-enabled interfaces are: • C sockets API (including C sockets for CICS) • Sockets Extended APIs: – Macro interface – Call Instruction interface (including Call Instruction for IMS and CICS)	

SMF Record Layout for FTP Client Calls

The SMF record written by FTP client calls has the following format:

Table 61. FTP Client SMF Record Format

Byte	Description
0–23	Standard SMF header with subtypes
24–27	FTP subcommand (RETR, STOR, or APPE)
28–31	Value of the reply to the FTP command
32–35	Client IP address
36–39	Server IP address

Table 61. FTP Client SMF Record Format (continued)

Byte	Description
40–41	Local port
42–43	Foreign port
44–47	Reserved
48–55	Remote userid
56	Data format (A—ASCII, E—EBCDIC, I—Image (binary), B—Double-byte, and U—UCS-2))
57	Transfer, S indicates stream data, and B indicates block data.
58	F indicates file structure, and R indicates record structure
59	Data set type (P—partitioned, blank—sequential, H—HFS)
60–63	Start time of transmission, if applicable, in hundredths of seconds
64–67	End time of transmission
68–71	Byte count if applicable
72	ID, this will be a C for client
76–119	Local data set name or PDS name. HFS data set names longer than 44 characters are truncated.
188–195	User ID of the user of FTP
196–203	Host ID
204–205	Offset to the first HFS file name field
206–207	Offset to the second HFS file name field

Two variable-length fields at the end of the record contain HFS file names. The variable-length HFS name fields have the following format:

Byte **Description**

0–1 Length of the HFS file name.

3–x HFS file name.

The value in the length field is the length of the HFS file name only; it does not include the two bytes for the length field itself. The maximum size of this variable field is 1025 bytes; the maximum value in the length subfield is 1023.

SMF Record Layout for Telnet Client Calls

The SMF record written by Telnet client calls has the following format:

Table 62. Telnet Client SMF Record Format

Byte	Description
0–23	Standard SMF header with subtypes
24–27	LGON or LGOF
44–45	Remote port address
46–47	Local port address
48–51	Remote IP address
52–55	Local IP address
56–63	Started task qualifier name
64–71	8-character NJE node name

SMF Record Layout for TCPIPSTATISTICS

Table 63. SMF Record Layout for TCPIPSTATISTICS

Off	Len	Name	Description
		Standard SMF Header	
00	1	SMFHDLEN	Record Length
02	2	SMFHDSEG	Segment descriptor
04	1	SMFHDFLG	Header flag byte
05	1	SMFHDTYP	Record type 118 (0x76)
06	4	SMFHDTME	Time when record was written
10	4	SMFHDDTE	Date when record was written
14	4	SMFHDSID	System identification
18	4	SMFHDSSI	Subsystem identification
20	2	SMFHDSUB	Record subtype (0x05)
22	2		Reserved
24	2	SMFHDSDL	Length of self-defining section
		Self-defining section	
26	4		Offset of subsystem section
30	2		Length of subsystem section
32	2		Number of subsystem sections (01)
34	4		Offset of IP section
38	2		Length of IP section
40	2		Number of IP sections (01)
42	4		Offset of ICMP section
46	2		Length of ICMP section
48	2		Number of ICMP sections (01)
50	4		Offset of TCP section
54	2		Length of TCP section
56	2		Number of TCP sections (01)
58	4		Offset of UDP section
62	2		Length of UDP section
64	2		Number of UDP sections (01)
		Subsystem Identification section	
0	8		TCP/IP Procname

Table 63. SMF Record Layout for TCPIPSTATISTICS (continued)

Off	Len	Name	Description
08	4		TCP/IP Asid
0C	8		TCP/IP Startup TOD
14	4		TCP/IP SMF Reason
14	1		x10 Last SMF record/Shutdown x20 Last SMF record/End stats x40 SMF Interval record x80 First SMF record
		IP Section	
00	4	imirecv	Total Received Datagrams
04	4	imihdrer	Total Discarded Datagrams
08	4	imiadrer	Total Discarded: Addr Errs
12	4	imifwddg	Total Attempt to Fwd Datagrams
16	4	imiunprt	Total Discarded: Unknown Prot
20	4	imidisc	Total Discarded: Other
24	4	imidelvr	Total Delivered Datagrams
28	4	imoreqst	Total Sent Datagrams
32	4	imodisc	Total Sent Discarded: Other
36	4	imonorte	Total Send Discarded: No Route
40	4	imrsmtos	Total Reassembly Timeouts
44	4	imrsmreq	Total Received: Reassem.Req'ed
48	4	imrsmok	Total Datagrams Reassembled
52	4	imrsmfld	Total Reassembly Failed
56	4	imfragok	Total Datagrams Fragmented
60	4	imfrgfld	Total Discarded: Frag. Failed
64	4	imfrgcre	Total Fragments Generated
68	4	imrtdisc	Total Routing Discards
72	4	imrsmmax	Max active Reassemblies
76	4	imrsmact	Num active Reassemblies
80	4	imrsmful	Discarding Reass fragments
		TCP Section	
0	4	RtoAlgorithm	Retransmit Algorithm
4	4	RtoMin	Min Retransmit Time (ms)
8	4	RtoMax	Max Retransmit Time (ms)
12	4	MaxConn	Max connections
16	4	ActiveOpens	Active Opens
20	4	PassiveOpens	Passive Opens
24	4	AttemptFails	Open failures

Table 63. SMF Record Layout for TCPIPSTATISTICS (continued)

Off	Len	Name	Description
28	4	EstabResets	Number of resets
32	4	CurrEstab	Num currently estabs
36	4	InSegs	Input Segments
40	4	OutSegs	Output Segments
44	4	RetransSegs	Retransmitted Segments
48	4	InErrs	Input Errors
52	4	OutRsts	Number of resets
		UDP Section	
0	4	usindgrm	Received UDP datagrams
4	4	usnoprts	UDP datagrams with no ports
8	4	usinerrs	Other UDP datagrams not rec'd
12	4	usotdgrm	UDP datagrams sent
		ICMP Section	
0	4	icmpInMsgs_	In ICMP Messages
4	4	icmpInErrors_	In ICMP Messages - Error
8	4	icmpInDestUnreachs_	In Destination Unreachable
12	4	icmpInTimeExcds_	In Time Exceeded Messages
16	4	icmpInParmProbs_	In Parameter Problem Messages
20	4	icmpInSrcQuenchs_	In Source Quench Messages
24	4	icmpInRedirects_	In Redirect Messages
28	4	icmpInEchos_	In Echo Request Messages
32	4	icmpInEchoReps_	In Echo Reply Messages
36	4	icmpInTimestamps_	In Timestamp Request Messages
40	4	icmpInTimestampReps_	In Timestamp Reply Messages *
44	4	icmpInAddrMasks_	In Address Mask Request Msgs
48	4	icmpInAddrMaskReps_	In Address Mask Reply Msgs
52	4	icmpOutMsgs_	Out ICMP Messages
56	4	icmpOutErrors_	Out ICMP Messages - Error
60	4	icmpOutDestUnreachs_	Out Destination Unreachable *
64	4	icmpOutTimeExcds_	Out Time Exceeded Messages
68	4	icmpOutParmProbs_	Out Parameter Problem Msgs
72	4	icmpOutSrcQuenchs_	Out Source Quench Messages
76	4	icmpOutRedirects_	Out Redirect Messages
80	4	icmpOutEchos_	Out Echo Request Messages
84	4	icmpOutEchoReps_	Out Echo Reply Messages
88	4	icmpOutTimestamps_	Out Timestamp Request Msgs

Table 63. SMF Record Layout for TCPIPSTATISTICS (continued)

Off	Len	Name	Description
92	4	icmpOutTimestampReps_	Out Timestamp Reply Messages
96	4	icmpOutAddrMasks_	Out Address Mask Request Msgs
100	4	icmpOutAddrMaskReps_	Out Address Mask Reply Msgs *

Appendix C. Creating a Keyring File for the Telnet Server

Note: For additional information about keyring files, refer to *OS/390 Cryptographic Services System Secure Sockets Layer Programming Guide and Reference*.

To use server authentication, the Telnet server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the Telnet server to the client application.

With server authentication, the Telnet server supplies the client with the Telnet server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure, encrypted communication channel is established between the Telnet server and the SSL enabled client.

For server authentication to work, the Telnet server must have a private key and associated server certificate in the server's key ring file and a password stash file.

To conduct commercial business on the Internet, you might use a widely known Certification Authority (CA), such as VeriSign, to get a high assurance server certificate. For a relatively small private network within your own enterprise or group, you can issue your own server certificates, called self-signed certificates, for your own use using the GSKKYMAN utility.

See "Using the GSKKYMAN Utility" for more information on the GSKKYMAN utility and requesting, receiving, and generating certificates.

The following high-level steps are required to enable SSL support for Telnet, with server authentication.

1. Generate the Telnet server private key and server certificate and stash file. See "Using the GSKKYMAN Utility" for more information.
2. Configure Telnet to include one or more SSL enabled ports (for example, SECUREPORT) and specify the name of the keyring file created in the step above.
3. Restart TCP/IP or issue Vary Obey with the updated configuration files.

The following sections provide more detail.

Using the GSKKYMAN Utility

Use the GSKKYMAN utility to create public/private key pairs and certificate requests, receive certificate requests into a key ring, and manage keys in a key ring. GSKKYMAN is a command-line utility. It prompts you for the information you need to perform a task. If you make an error, it issues a message and prompts you again for the information.

GSKKYMAN is documented in *OS/390 Cryptographic Services System Secure Sockets Layer Programming Guide and Reference*.

To run GSKKYMAN, you must have access to the OS/390 Cryptographic Services message catalogs and DLLs. For example, if the OS/390 Cryptographic Service DLL library is not part of the linklist concatenation, an ″export STEPLIB=hlq.SGSKLOAD″ command might be needed. For additional information, refer to *OS/390 Cryptographic Services System Secure Sockets Layer Programming Guide and Reference*.

MKKF format keyring files can be converted to GSKKYMAN format files using the -m option of the GSKKYMAN command. To do this, enter gskkyman -m, and you will be prompted for the name of the MKKF format file. For more information, refer to *OS/390 SecureWay Communications Server: IP Migration*.

Where to Find GSKKYMAN

GSKKYMAN is shipped with OS/390 in System SSL which is part of Cryptographic Services Base element of OS/390 (FMID HCPT270). This version of GSKKYMAN supports a key size up to 512 bits. To obtain a GSKKYMAN that supports larger keys (for example, the Strong Crypto support for OS/390 Cryptographic Services), obtain one of the following products:

- Security Level 2 Cryptographic Services (FMID JCPT283)
- Security Level 3 Cryptographic Services (FMID JCPT271)

Using GSKKYMAN with the Telnet Server

GSKKYMAN runs under the OS/390 shell and creates the following several HFS files:

- A keyring file
- A stash file, which contains the password associated with the keyring file
- A certificate file

The keyring file and the stash file are used by the Telnet server to obtain the server's certificate and the public/private key pair used during SSL handshake processing. The Telnet server uses the stash file as the mechanism to obtain the keyring password rather than using a configuration parameter which might be accessible to a larger number of people.

Security of these files is an installation responsibility. It is recommended to restrict the file access to users with superuser authority.

MVS files can be created from the HFS files by using the TSO OGET command with the BINARY option and can be protected using RACF. For example:

```
OGET '/tmp/telnet/mvs180.kdb' 'TCPCS6.MVS180.KDB' BINARY
OGET '/tmp/telnet/mvs180.sth' 'TCPCS6.MVS180.STH' BINARY
```

It is recommended that the MVS dsnames be the HFS filenames prefixed by one or more high-level qualifiers. The same high-level qualifier(s) must be used for both the keyring and the stash file. This ensures that the name relation used to generate the stash file from the keyring file is unchanged. Refer to *OS/390 OpenEdition Command Reference* for more information on the use of the OGET command. The MVS keyring and stash files can be used by the Telnet server.

Note: GSKKYMAN only accepts the HFS files.

GSKKYMAN allows you to enter the fully-qualified path and file name when it prompts you for a keyring, certificate request, or certificate file name. However, you should change to the path where the file should be stored before you start GSKKYMAN.

Self-signed Server Certificates

Normally, a server certificate should be obtained from a known Certificate Authority (CA), and the client has likely been primed with the well known CA. However, for testing, an installation might use a self-signed server certificate. Because the clients will not know about the issuer of the self-signed server certificate, in most cases it is necessary to add the server's self-signed certificate to the client's signer certificates (HOD and PCOMM maintain this list in CustomizedCAs.class).

An example of how this is done for a HOD NT client follows:

- On the TN3270 server:
  1. Using gskkyman, select 'Create a self-signed certificate'. Specify:
     a. Version 3
     b. Set the key as the default in your key database.
     c. Save the certificate to a file, select binary format (the certificate will be saved in binary DER format).
  2. Using binary mode, ftp in the binary DER certificate created in step 3 above to the client.
- On the client side using HOD's certificate management utility:
  1. If customized CA certificates have previously been added to HOD , open the CustomizedCAs.class file, or if this is the first customized CA, create a new class file by:
     – Selecting FILE, then New.
     – Click on the 'file data base type ' arrow and select 'SSLight key database class'. This will fill in the required filename and path
  2. Select 'ADD'.
  3. Select data type 'Binary DER data'.
  4. Specify the binary certificate file that was ftped from the TN3270 server above.
  5. Close the file after completing the ADD.
- Restart HOD and TCP to pick up the certificates that have been added to the respective databases.

Self-signed Client Certificates

Normally, a client certificate should be obtained from a known Certificate Authority (CA). The Certificate Authority's root certificate needs to be included in TN3270 server's key data base in order for the clients certificate to pass the SSL protocol's client authentication process.

If an installation uses self-signed client certificates for testing purposes, each certificate appears to be issued by a unique CA and a CA certificate for each client certificate must be entered into the server's key data base.

Steps to perform this vary depending on the source of the client certificate. Two examples follow:

- Self-signed certificate created by gskkyman
- Self-signed certificate issued by HOD's certificate management utility

Self-signed Certificate Created by GSKKYMAN

1. On the TN3270 server, use GSKKYMAN to:
   - Create a self-signed certificate
   - Create a PKCS12 format file (select 'Export keys', 'Export keys to a PKCS12 file' from the gskkyman panels). This will create a file with the format filename.p12. If using CLIENTAUTH SAFCERT, use this file as the client certificate source if manually registering the client certificate to the SAF product
2. FTP the PKCS12 file in binary mode to the client.
3. Use the HOD certificate management panels to import the PKCS12 file into the HOD key data base.
4. Export the certificate using the HOD certificate management panels to create a new PKCS12 file. This is the file that HOD will use to retrieve the client certificate.

   Note: This step is needed because gskkyman generates a version 1 PKCS12 file. HOD expects a version 3 PKCS12 file when doing the SSL handshake. The HOD certificate management utility can read both formats, but creates version 3 PKCS12 files.

Self-signed Certificate Created by HODs key Management Utility

1. On the HOD client, use HOD's certificate management utility to:
   - Create a self-signed certificate.
   - Use the export function to create a PKCS12 file. This is the file that the HOD client will use.
   - Use the extract function to create a binary DER data file. This file will have the format filename.der
2. FTP the binary DER data file to the TN3270 server in binary format.
3. On the TN3270 server, use gskkyman to store the .der file using the 'Store a CA certificate' option.
4. If using CLIENTAUTH SAFCERT, use the filename.der file as the source for manually registering the client certificate to the SAF product.

   Note: RACF certificate registration requires an MVS data set; therefore, you must first create an MVS file using the TSO OGET command. For example:

   ```
   OGET  '/tmp/telnet/clntcert.der'  'TCPCS.CLNTCERT.DER'   BINARY
   ```

Appendix D. Related Protocol Specifications (RFCs)

This appendix lists the related protocol specifications for TCP/IP. The internet protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

Many features of TCP/IP for MVS are based on the following RFCs:

RFC	Title and Author
768	*User Datagram Protocol* J.B. Postel
791	*Internet Protocol* J.B. Postel
792	*Internet Control Message Protocol* J.B. Postel
793	*Transmission Control Protocol* J.B. Postel
821	*Simple Mail Transfer Protocol* J.B. Postel
822	*Standard for the Format of ARPA Internet Text Messages* D. Crocker
823	*DARPA Internet Gateway* R.M. Hinden, A. Sheltzer
826	*Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware* D.C. Plummer
854	*Telnet Protocol Specification* J.B. Postel, J.K. Reynolds
855	*Telnet Option Specification* J.B. Postel, J.K. Reynolds
856	*Telnet Binary Transmission* J.B. Postel, J.K. Reynolds
857	*Telnet Echo Option* J.B. Postel, J.K. Reynolds
858	*Telnet Suppress Go Ahead Option* J.B. Postel, J.K. Reynolds
859	*Telnet Status Option* J.B. Postel, J.K. Reynolds
860	*Telnet Timing Mark Option* J.B. Postel, J.K. Reynolds
861	*Telnet Extended Options —List Option* J.B. Postel, J.K. Reynolds
862	*Echo Protocol* J.B. Postel
863	*Discard Protocol* J.B. Postel
864	*Character Generator Protocol* J.B. Postel
877	*Standard for the Transmission of IP Datagrams over Public Data Networks* J.T. Korb
885	*Telnet End of Record Option* J.B. Postel
903	*Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
904	*Exterior Gateway Protocol Formal Specification* D.L. Mills
919	*Broadcasting Internet Datagrams* J.C. Mogul
922	*Broadcasting Internet Datagrams in the Presence of Subnets* J.C. Mogul

950 *Internet Standard Subnetting Procedure* J.C. Mogul, J.B. Postel

952 *DoD Internet Host Table Specification* K. Harrenstien, M.K. Stahl, E.J. Feinler

959 *File Transfer Protocol* J.B. Postel, J.K. Reynolds

974 *Mail Routing and the Domain Name System* C. Partridge

1006 *ISO Transport Service on tiop of the TCP Version 3* M.T.Rose, D.E. Cass

1009 *Requirements for Internet Gateways* R.T. Braden, J.B. Postel

1013 *X Window System Protocol, Version 11: Alpha Update* R.W. Scheifler

1014 *XDR: External Data Representation Standard* Sun Microsystems Incorporated

1027 *Using ARP to Implement Transparent Subnet Gateways* S. Carl-Mitchell, J.S. Quarterman

1032 *Domain Administrators Guide* M.K. Stahl

1033 *Domain Administrators Operations Guide* M. Lottor

1034 *Domain Names—Concepts and Facilities* P.V. Mockapetris

1035 *Domain Names—Implementation and Specification* P.V. Mockapetris

1042 *Standard for the Transmission of IP Datagrams over IEEE 802 Networks* J.B. Postel, J.K. Reynolds

1044 *Internet Protocol on Network System's HYPERchannel: Protocol Specification* K. Hardwick, J. Lekashman

1055 *Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP* J.L. Romkey

1057 *RPC: Remote Procedure Call Protocol Version 2 Specification* Sun Microsystems Incorporated

1058 *Routing Information Protocol* C.L. Hedrick

1073 *Telnet Window Size Option* D. Waitzman

1079 *Telnet Terminal Speed Option* C.L. Hedrick

1091 *Telnet Terminal-Type Option* J. VanBokkelen

1094 *NFS: Network File System Protocol Specification* Sun Microsystems Incorporated

1096 *Telnet X Display Location Option* G. Marcy

1112 *Host Extensions for IP Multicasting* S. Deering

1118 *Hitchhikers Guide to the Internet* E. Krol

1122 *Requirements for Internet Hosts—Communication Layers* R.T. Braden

1123 *Requirements for Internet Hosts—Application and Support* R.T. Braden

1155 *Structure and Identification of Management Information for TCP/IP-Based Internets* M.T. Rose, K. McCloghrie

1156 *Management Information Base for Network Management of TCP/IP-Based Internets* K. McCloghrie, M.T. Rose

1157 *Simple Network Management Protocol (SNMP)* J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin

1179 *Line Printer Daemon Protocol* The Wollongong Group, L. McLaughlin III

1180 *TCP/IP Tutorial* T.J. Socolofsky, C.J. Kale

1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris, (Updates RFC 1034, RFC 1035)

1184 *Telnet Linemode Option* D. Borman

1187 *Bulk Table Retrieval with the SNMP* M.T. Rose, K. McCloghrie, J.R. Davin

1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

1191 *Path MTU Discovery* J. Mogul, S. Deering

1198 *FYI on the X Window System* R.W. Scheifler

1207 *FYI on Questions and Answers: Answers to Commonly Asked "Experienced Internet User" Questions* G.S. Malkin, A.N. Marine, J.K. Reynolds

1208 *Glossary of Networking Terms* O.J. Jacobsen, D.C. Lynch

1213 *Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II* K. McCloghrie, M.T. Rose

1215 *Convention for Defining Traps for Use with the SNMP* M.T. Rose

1228 *SNMP-DPI Simple Network Management Protocol Distributed Program Interface* G.C. Carpenter, B. Wijnen

1229 *Extensions to the Generic-Interface MIB* K. McCloghrie

1230 *IEEE 802.4 Token Bus MIB IEEE 802 4 Token Bus MIB* K. McCloghrie, R. Fox

1231 *IEEE 802.5 Token Ring MIB IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker

1267 *A Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter

1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross

1269 *Definitions of Managed Objects for the Border Gateway Protocol (Version 3)* S. Willis, J. Burruss

1270 *SNMP Communications Services* F. Kastenholz, ed.

1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

1325 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G.S. Malkin, A.N. Marine

1340 *Assigned Numbers* J.K. Reynolds, J.B. Postel

1348 *DNS NSAP RRs* B. Manning

1350 *TFTP Protocol* K.R. Sollins

1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

1354 *IP Forwarding Table MIB* F. Baker

1356 *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann

1372 *Telnet Remote Flow Control Option* D. Borman, C. L. Hedrick

1374 *IP and ARP on HIPPI* J. Renwick, A. Nicholson

1381 *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker

1382 *SNMP MIB Extension for the X.25 Packet Layer* D. Throop

1387 *RIP Version 2 Protocol Analysis* G. Malkin

1388 *RIP Version 2 — Carrying Additional Information* G. Malkin

1389 *RIP Version 2 MIB Extension* G. Malkin

1390 *Transmission of IP and ARP over FDDI Networks* D. Katz

1393 *Traceroute Using an IP Option* G. Malkin

1397 *Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol* D. Haskin

1398 *Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz

1540 *IAB Official Protocol Standards* J.B. Postel

1571 *Telnet Environment Option Interoperability Issues* D. Borman

1572 *Telnet Environment Option* S. Alexander

1577 *Classical IP and ARP over ATM* M. Laubach

1583 *OSPF Version 2* J. Moy

1592 *Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters

1594 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* A.N. Marine, J. Reynolds, G.S. Malkin

1695 *Definitions of Managed Objects for ATM Management Version 8.0 Using SMIv2* M. Ahmed, K. Tesink

1723 *RIP Version 2 — Carrying Additional Information* G. Malkin

1850 *OSPF Version 2 Management Information Base* F. Baker, R. Coltun

1901 *Introduction to Community-Based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1902 *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1903 *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1904 *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1905 *Protocols Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1908 *Coexistence between Version 1 and Version 2 of the Internet-Standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1939 *Post Office Protocol-Version 3* J. Myers, M. Rose

2011 *SNMPv2 Management Information Base for the Internet Protocol Using SMIv2* K. McCloghrie

2012 *SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2* K. McCloghrie

2013 *SNMPv2 Management Information Base for the User Datagram Protocol Using SMIv2* K. McCloghrie

2205 *Resource ReSerVation Protocol (RSVP) Version 1*R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin

2233 *The Interfaces Group MIB Using SMIv2* K. McCloghrie, F. Kastenholz

2271 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2272 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2273 *SNMP Applications* David B. Levi, Paul Meyer, Bob Stewart

2274 *User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2275 *View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2320 *Definitions of Managed Objects for Classical IP and ARP over ATM Using SMIv2* M. Greene, J. Luciani, K. White, T. Kuo

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 by subscription. Online copies are available using FTP from the NIC at the following Web address: http://www.rfc-editor.org/rfc.html

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`.

Appendix E. Syslog Daemon

Syslog daemon (syslogd) is a server process that must be started as one of the first processes in your OS/390 UNIX environment. CS for OS/390 server applications and components use syslogd for logging purposes and can also send trace information to syslogd. Servers on the local system use AF_UNIX sockets to communicate with syslogd; remote servers use the AF_INET socket. CS for OS/390 components use the **local1** and **daemon** facility names.

Note: Each application activates and deactivates traces in a slightly different manner. For details, refer to the chapter on the individual application.

The syslog daemon reads and logs system messages to the MVS console, log files, other machines, or users as specified by the configuration file. If syslogd is not started, application log data may appear on the MVS console. Figure 56 shows how syslogd operates in the OS/390 UNIX environment.

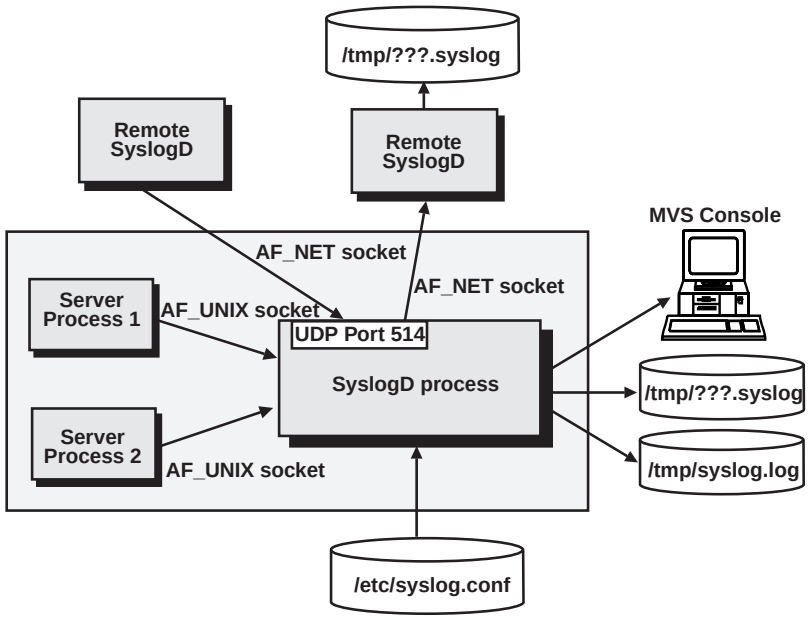


Figure 56. Syslogd Operation

The configuration file is read at startup and whenever the hang-up signal (SIGHUP) is received. The syntax for configuration statements is described in “Configuration Statements” on page 1185.

The syslog daemon stores its process ID in the /etc/syslog.pid file so that it may be used to terminate or reconfigure the daemon. In order for syslogd to successfully create this file, you must define the syslogd user ID as UID=0.

Messages are read from the UNIX domain datagram socket and the Internet domain datagram socket. Kernel messages are not logged in OS/390 UNIX MVS.

Note: For an example of how syslogd can be used in your environment, refer to *OS/390 eNetwork Communications Server V2R5 TCP/IP Implementation Guide, Volume 2: OpenEdition Applications* and *Accessing OS/390 OpenEdition from the Internet*.

Files

The syslog daemon uses the following files:

/dev/console
Operator console

/etc/syslog.pid
Location of the process ID

/etc/syslog.conf
Default configuration file

/dev/log
Default log path for UNIX datagram socket

/usr/sbin/syslogd
Server

Starting syslogd

Following is the syntax for the syslogd command:

syslogd [−f *conffile***] [−m** *markinterval***] [−p** *logpath***]**

syslogd recognizes the following options:

-f Configuration file name

-m Number of minutes between mark messages

-p Path name for the OS/390 UNIX datagram socket

- If you want ITRACE messages from TCP/IP initialization written to syslogd and do not want trace messages from TCP/IP or inetd written to the master console, you must start syslogd before TCP/IP and inetd:
  1. Start syslogd
  2. Start TCP/IP
  3. Start inetd
- For special considerations on remote syslogd servers, see “Usage Notes” on page 1188.
- To specify the jobname and pass the appropriate environment variables to the syslogd process, start syslogd using a shell script such as the following:

```
#
# Start the syslog daemon
#

export _BPX_JOBNAME='SYSLOGD'
/usr/sbin/syslogd -f /etc/syslog.conf &

echo -- /u/silviar/syslogd.sh script executed,  date
```

You can execute this shell script directly from the /etc/rc file to start syslogd at OS/390 UNIX initialization.

Operation

- To periodically offload log files to another location and delete unwanted messages without stopping syslogd, follow these steps:
  1. Create two syslogd configuration files called /etc/syslog.conf.a and /etc/syslog.conf.b. These two files will be identical except that all log files end with either “a” or “b.”
  2. If your current /etc/syslog.conf file was created from your syslog.conf.a file, copy your syslog.conf.b file to /etc/syslog.conf.
  3. Send syslogd a SIGHUP signal. This stops syslogd from writing to the “a” log files and forces it to write to the “b” log files.
  4. Offload the “a” files and delete their current contents.
- To terminate syslogd, send a SIGTERM signal.

  ```
  kill -s SIGTERM <PID>
  ```
- To force syslogd to reread its configuration file and activate any modified parameters without stopping, send a SIGHUP signal. Syslogd will continue to append log messages to the files you specify in /etc/syslog.conf.

  ```
  kill -s SIGHUP <PID>
  ```
- If an invalid argument or number of arguments is entered, syslogd exits and the return code is 1. In all other situations in which syslogd exits, the return code is 0 (zero).

Configuration Statements

The processing of syslogd is controlled by a configuration file called /etc/syslog.conf (see Figure 57 on page 1186 for a sample file) in which you define logging rules and output destinations for error messages, authorization violation messages, and trace data. Logging rules are defined using a *facility* name and a *priority* code. The *facility* name and *priority* code are passed on the logging request from an application when it wants to log a message.

```
#
# facility-name.priority  destination
# ----------------------  -----------
#
# All alert messages (and above
# priority messages) go to the
# MVS console
#
*.alert                /dev/console
#
#
# All authorization messages
# go to auth.log
#
auth.*                 /tmp/syslogd/auth.log
#
# All error messages (and
# above priority messages) go
# to error.log
#
*.err                  /tmp/syslogd/error.log
#
# All debug messages (and
# above priority messages) from
# telnet go to telnet.debug
#
local1.debug          /tmp/syslogd/telnet.debug
#
# All ftpd, rexecd, rshd, SNMP agent and SNMP subagent
# debug messages (and above
# priority messages) go
# to server.debug
#
daemon.debug          /tmp/syslogd/server.debug
#
# Everything not directed to
# a destination above, is directed
# to a garbagecan.log (so nothing important
# is lost)
#
*.*;local1.none;daemon.none;auth.none /tmp/syslogd/garbagecan.log
```

Figure 57. Sample /etc/syslog.conf File

Configuration Notes:

- Comments can be added to the configuration file by placing the # character in column one of the comment line. Everything following the hashmark (#) character will be treated as a comment.
- When you specify a priority code, all messages with that priority *and higher* are logged at the specified destination. For example, if you specify a priority code of crit, all messages having alert, panic, emerg, and crit priorities are logged. To send all messages with a priority of crit or higher to a user ID of OPER1, you can enter the following rule in /etc/syslog.conf:

  ```
  *.crit  OPER1
  ```

- You can combine logging rules and destinations in different ways. For example, to send all messages from the facility name daemon into one file and all messages with a priority of crit or higher into another file, enter the following:

  ```
  daemon.*  /tmp/syslogd/daemon.log
  *.crit    /tmp/syslogd/crit.log
  ```

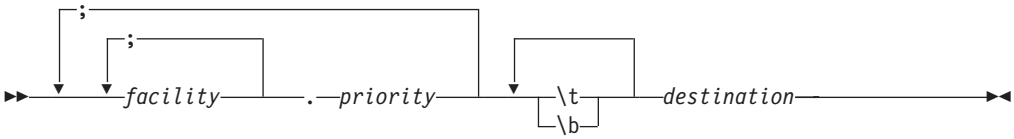
Note that if a server sends a message to syslogd with a facility name of daemon and a priority code of crit, messages will be logged in both the daemon.log and crit.log files. Likewise, if a server sends a message to syslogd with a facility name of daemon and a priority code of alert, the message will be logged in both files.

- A priority code of none tells syslogd not to select any messages. For example, if you want to log all messages from facility name local1 into one file, all messages from the daemon into another file, and all remaining messages into a third file, use the following:

```
local1.*                        /tmp/syslogd/local1.log
daemon.*                        /tmp/syslogd/daemon.log
*.*;local1.none;daemon.none     /tmp/syslogd/the_rest.log
```

- You cannot define logging conditions related to a jobname, process name, or pid. All messages that belong to the same facility or priority class are logged in the same syslogd logging file, whether or not the server task has issued the message.

Each statement of the configuration file has the following syntax:



Facility Names

The following facility names are supported and predefined in the syslogd implementation:

user Message generated by a process (user)

mail Message generated by mail system

news Message generated by news system

uucp Message generated by UUCP system

daemon
This facility name is generally used by server processes. The FTPD server, the RSHD server, the REXECD server, the SNMP agent, and the SNMP subagent use this facility name to log trace messages.

auth/authpriv
Message generated by authorization daemon

cron Message generated by the clock daemon

lpr Message generated by printer system

local0-7
Names for local use. The OS/390 UNIX Telnet server uses the local1 facility name for its log messages.

mark Used for logging MARK messages

Priority Codes

The following priority codes are supported. They are shown in priority sequence.

emerg/panic A panic condition was reported to all processes.

alert A condition that should be corrected immediately.

crit A critical condition

err(or) An error message

warn(ing) A warning message

notice A condition requiring special handling

info A general information message

debug A message useful for debugging programs

none Do not log any messages for the facility

***** Place holder used to represent all priorities

The \t in the syntax diagram is a tab character; the \b is a blank space.

Destinations

The following destinations are supported. Be sure to use lowercase for all file names, users, and hosts:

/file A specific file (for example, /tmp/syslogd/auth.log). All log files used by syslogd must be created in the hierarchical file system (HFS) before syslogd is started.

@host A syslog daemon on another host (for example, @mya1xserver).

user1,user2,...
A list of users

Note: Be careful with this option. If the user is not logged on, all messages will be queued in OS/390 UNIX, causing it to crash.

To send messages to all logged-on users, use an asterisk as a “user name”:

```
facility_name.priority_code  *
```

/dev/console The MVS console

Usage Notes

- syslogd can only be started by a superuser.
- syslogd can be terminated using the SIGTERM signal.
- If you want syslogd to receive log data from or send log data to remote syslogd servers, reserve UDP port 514 for the syslogd job in your PROFILE.TCP/IP data set and enter the syslog service for UDP port 514 in the services file or dataset (for example, /etc/services/). The following example shows the PROILE.TCP/IP entry:

```
PORT
...
514 UDP ETCINIT3     ;syslogd daemon
...
```

The following example shows the /etc/services file entry:

```
syslog     514/udp
```

Note: If you do not want syslogd to communicate with remote syslogd servers, remove the syslog service from the services file or dataset (for example, /etc/services/).

- If there is no TCP/IP transport active when syslogd starts or if TCP/IP is recycled, syslogd will establish or reestablish communication with TCP/IP when it becomes available.
- Configuration file errors are written to the operator console because initialization is not complete until the entire configuration file has been read.
- Facility mark is not affected by the *.priority usage.
- The minimum mark interval is 30 seconds.

Application Considerations

You can use the logging facilities of the syslogd server with your OS/390 UNIX application programs. Include the syslog.h header file with C programs so that they can open a log facility, send log messages to syslogd, and close the facility:

```
#include <syslog.h>

1 openlog("oec", LOG_PID, LOG_LOCAL0);
2 syslog(LOG_INFO, "Hello from oec");
3 closelog();
```

1 Open a log facility with the name of local0. Prefix each line in the log file with the program name (oec) and the process ID.

2 Log an info priority message with the specified content.

3 Close the log facility name.

The preceding statements created the following line in the log file:

```
May 26 11:27:51 mvs18oe oec[3014660]: Hello from oec
```

For more information about the syslog function, refer to *Advanced Programming in the UNIX Environment*, published by Addison-Wesley or *OS/390 C/C++ IBM Open Class Library Reference*.

Appendix F. Setting up the inetd Configuration File

inetd is a generic listener program used by such servers as OS/390 UNIX telnet server and OS/390 UNIX rexec server. Other servers such as OS/390 UNIX ftp server have their own listener program and do not use inetd.

inetd.conf is an example of the user's configuration file. It is stored in the /etc directory. Ensure that the inetd services required on your system are enabled using configuration statements like those in the following example:

```
#=====================================================================
# service | socket | protocol | wait/ | user | server   | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
 shell    stream    tcp        nowait  OMVSKERN /usr/sbin/orshd rshd -l
 exec     stream    tcp        nowait  OMVSKERN /usr/sbin/orexecd rexecd -LV
 otelnet  stream    tcp        nowait  OMVSKERN /usr/sbin/otelnetd otelnetd  -LV
```

Figure 58. Adding Applications to /etc/inetd.conf

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the OpenEdition environment, insure that statements have been added to ETC.SERVICES for each of these servers. See the sample ETC.SERVICES installed in the /usr/lpp/tcpip/samples/services directory for how to specify ETC.SERVICES statements for these servers.

The traces for both the OE REXECD server and the OE RSHD server are enabled via options in the inetd configuration file (/etc/inetd.conf):
The traces are turned on for both servers by passing a -d argument to the server

```
#=====================================================================
# service | socket | protocol | wait/ | user | server   | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
shell      stream    tcp       nowait OMVSKERN /usr/sbin/orshd rshd -d  1
exec       stream    tcp       nowait OMVSKERN /usr/sbin/orexecd rexecd -d  2
```

Figure 59. Setting Traces in /etc/inetd.conf

programs. **1** is the RSHD server and **2** is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and the inetd server has reread the file will produce trace output.

The trace is written in formatted form to the syslogd facility name daemon with a priority of debug. The trace data can be routed to a file in your Hierarchical File System by specifying the following definition in your syslogd configuration file (/etc/syslogd.conf):

```
  #
  # All ftp, rexecd, rshd
  # debug messages (and above
  # priority messages) go
# to server.debug.a
```

```
#
daemon.debug              /tmp/syslogd/server.debug.a
```

In this example, the trace data is written to /tmp/syslogd/daemon.debug.a in your Hierarchical File System. For more information on syslogd, refer to “Appendix E. Syslog Daemon” on page 1183.

For more information about inetd, refer to *OS/390 UNIX System Services Planning*, SC28-1890-03 or *Accessing OS/390 OpenEdition MVS from the Internet* (SG24-4721).

Appendix G. Protocol Number and Port Assignments

This appendix provides protocol-number assignments and test configuration port assignments used in the PROFILE.TCPIP configuration data set and the HFS file /etc/services.

Protocol Assignments from /etc/protocol

The HFS file, /etc/protocol, specifies protocol numbers. Following is a sample. If you have an installation with a customized /etc/protocol file, ensure that OSPF is set to 89.

```
# @(#)protocols 7.2 88/06/28 12:40:03
#
# /etc/protocols file for AIX
#
# official name, protocol number, aliases

ip              0               # dummy for IP
icmp            1               # control message protocol
# ggp           2               # gateway¬2 (deprecated)
tcp             6               # tcp
# egp           8               # exterior gateway protocol
# pup           12              # pup
udp             17              # user datagram protocol
# idp           22              # xns idp
ospf            89              # Open Shortest Path First protocol
```

Port Assignments

Port numbers are used on various socket calls. They are also included in both the header of a TCP segment and a UDP datagram. You can assign port numbers to your own server applications by adding entries to the HFS file or to the data set.

The recommended convention in assigning ports is to assign standard port numbers for TSO procedures and port numbers beginning with ″2″ for the corresponding OS/390 UNIX daemons. For example, the RXSERVE procedure is assigned to ports 512 and 514, and the orexecd and orshd daemons are assigned to ports 2512 and 2514. Similarly, TSO TELNET is assigned to port 23, and otelnetd is assigned to port 2023. The MISCSERV procedure is assigned to ports 7, 9, and 19. The echo, discard and chargen applications are assigned to ports 2007, 2009 and 2019 respectively.

PROFILE.TCPIP Port Assignments

Use the PORT statement in the PROFILE.TCPIP data set to reserve ports for specified user IDs, procedures, and job names. The following example was used for test configuration and is for illustration only. The shipped data set contains the most current assignments.

```
     7 UDP MISCSERV            ; Misc server
     7 TCP MISCSERV            ; Misc server
     9 UDP MISCSERV            ; Misc server
     9 TCP MISCSERV            ; Misc server
    13 TCP OMVS                ; OMVS Daytime
    13 UDP OMVS                ; OMVS Daytime
    19 UDP MISCSERV            ; Misc server
    19 TCP MISCSERV            ; Misc server
    20 TCP OMVS                ; Ftpd
```

```
   21 TCP OMVS                ; Ftpd
   23 TCP INTCLIEN            ; Telnet
   25 TCP SMTP                ; SMTP server
   53 TCP NAMESRV             ; Name Server
   53 UDP NAMESRV             ; Name Server
  111 UDP OMVS                ; OPortmap Server
  111 TCP OMVS                ; OPortmap Server
  135 UDP LLBD                ; NCS
  161 UDP OSNMPD              ; SNMP Agent
  162 UDP OMVS                ; SNMP Client for receipt of traps
  512 TCP RXSERVE             ; Remote Execution Server
  514 TCP RXSERVE             ; Remote Shell Server
  513 UDP OMVS                ; Remote Login
  515 TCP LPDSERVE            ; LPD server
  520 UDP OMVS                ; ORouteD Server
  529 UDP OMVS                ; SyslogD Server
  750 TCP MVSKERB             ; Kerberos server
  750 UDP MVSKERB             ; Kerberos server
  751 TCP ADM@SERV            ; Kerberos server
  751 UDP ADM@SERV            ; Kerberos server
 2023 TCP OMVS                ; Telnetd
 2512 UDP OMVS                ; Remote Execution Server
 2512 TCP OMVS                ; Remote Execution Server
 2514 TCP OMVS                ; Remote Shell Server
 2514 UDP OMVS                ; Remote Shell Server
 3010 TCP CICS510             ; CICS Listener V5R1
 3011 TCP CICS510             ; CICS Listener V5R1
 3012 TCP CICS510             ; CICS Listener V5R1
 3013 TCP CICS510             ; CICS Listener V5R1
 3014 TCP CICS510             ; CICS Listener V5R1
 3015 TCP CICS510             ; CICS Listener V5R1
 4000 TCP EZAIMSLN            ; IMS Listener V5R1.0
```

/etc/services Port Assignments

The HFS file, /etc/services, contains the port assignments of specific OS/390 UNIX applications. The following example was used for test configuration and is for illustration only. The shipped file contains the most current assignments.

```
#
# 5799-WZQ (C) COPYRIGHT IBM CORPORATION 1987,1988
# LICENSED MATERIALS - PROPERTY OF IBM
# REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
#
# $Header:services 9.4$
# $ACIS:services 9.4$
# $Source: /ibm/acis/usr/src/etc/RCS/services,v $

#       @(#)services    1.16 (Berkeley) 86/04/20
#
# Network services, Internet style
#
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp             21/tcp
telnet          23/tcp
```

```
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
tftp            69/udp
rje             77/tcp          netrjs
finger          79/tcp
link            87/tcp          ttylink
supdup          95/tcp
hostnames       101/tcp         hostname        # usually from sri-nic
#csnet-cs       105/?
pop             109/tcp         postoffice
sunrpc          111/tcp
sunrpc          111/udp
auth            113/tcp         authentication
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
snmp            161/udp                         # snmp request port
snmp-trap       162/udp                         # snmp monitor trap port
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
efs             520/tcp                         # for LucasFilm
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem

ingreslock      1524/tcp
#
# Start of IBM added services ...
#
route           520/udp         router routed
ncprout         580/udp         ncproute

#
# RVD service
#
rvd-control     531/udp                         # rvd control port
#
# Andrew File System services
#
filesrv         2001/tcp
console         2018/udp
venus.itc       2106/tcp

# For file server backup and migration
client          2030/tcp
```

```
#
# Andrew File System Authenticated services
#
vexec           712/tcp         vice-exec
vlogin          713/tcp         vice-login
vshell          714/tcp         vice-shell

# For the Venus process.
venus.itc       2106/tcp
rauth2          2001/udp
rfilebulk               2002/udp
rfilesrv                2003/udp
ropcons         2115/udp
# The following are assigned in pairs and the bulk must be the srv +1
rupdsrv         2131/udp
rupdbulk                2132/udp
rupdsrv1                2133/udp
rupdbulk1               2134/udp

#
#     Kerberos services
#

klogin          543/tcp           # Kerberos aythenticated rlogin
kerberos        750/udp      kdc  # Kerberos aythentication--udp
kerberos        750/tcp      kdc  # Kerberos aythentication--tcp
kerboros_master 751/udp           # Kerberos aythentication
kerberos_master 751/tcp           # Kerberos aythentication
passwd_server   752/udp           # Kerberos passwd server
userreg_server  753/tcp           # Kerberos userreg serrver
kpop            1109/tcp          # Pop with Kerberos
knetd         2053/tcp                # Kerberos de-multiplexor
kshell          544/tcp         cmd     # and remote shell
eklogin         2105/tcp                # Kerberos encrypted rlogin
krb_prop        754/tcp                 # Kerboros slpave propagation
erlogin         888/tcp                 # Login and environment passing
#
#
#       Kerberos sample server
#
sample        906/tcp                # Kerberos sample app server
sample                        906/udp               #for kerberos simple test
```

Appendix H. How to Read a Syntax Diagram

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and Punctuation

The following symbols are used in syntax diagrams:

Symbol	Description
►►	Marks the beginning of the command syntax.
►	Indicates that the command syntax is continued.
|	Marks the beginning and end of a fragment or part of the command syntax.
►◄	Marks the end of the command syntax.

You must include all punctuation such as colons, semicolons, commas, quotation marks, and minus signs that are shown in the syntax diagram.

Parameters

The following types of parameters are used in syntax diagrams.

Required
Required parameters are displayed on the main path.

Optional
Optional parameters are displayed below the main path.

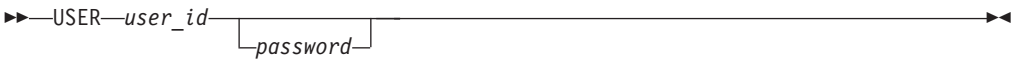
Default
Default parameters are displayed above the main path.

Parameters are classified as keywords or variables. Keywords are displayed in uppercase letters and can be entered in uppercase or lowercase. For example, a command name is a keyword.

Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

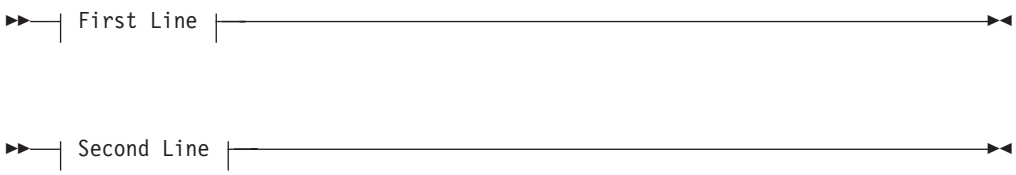
Syntax Examples

In the following example, the USER command is a keyword. The required variable parameter is *user_id*, and the optional variable parameter is *password*. Replace the variable parameters with your own values.



Longer Than One Line

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.



Required Operands

Required operands and values appear on the main path line.

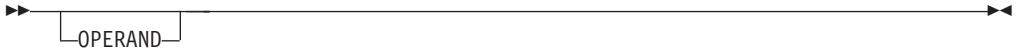
You must code required operands and values.

Choose One Required Item from a Stack

If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically in alphanumeric order.

Optional Values

Optional operands and values appear below the main path line.



You can choose not to code optional operands and values.

Choose One Optional Operand from a Stack

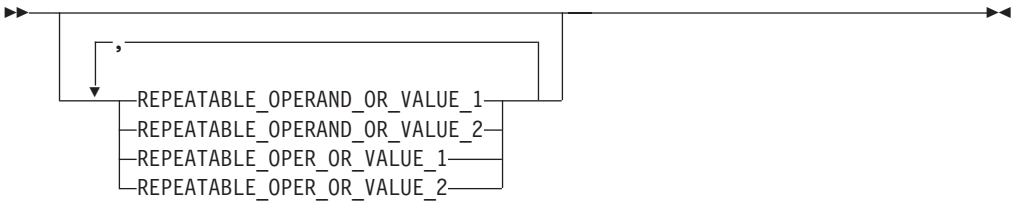
If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically in alphanumeric order below the main path line.

Repeating an Operand

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

Selecting More Than One Operand

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



If an operand or value can be abbreviated, the abbreviation is described in the text associated with the syntax diagram.

Nonalphanumeric Characters

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

Blank Spaces in Syntax Diagrams

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

Default Operands

Default operands and values appear above the main path line. TCP/IP uses the default if you omit the operand entirely.

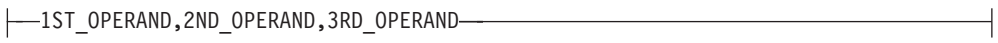
Variables

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

Syntax Fragments

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

Syntax Fragment:



Appendix I. Information Apars

This appendix lists information apars for IP-related books.

Note: Information apars contain updates to previous editions of the manuals listed below. Books updated for V2R8 contain all the updates except those contained in the information apars that may be issued after V2R8 books went to press.

IP Information Apars

Table 64 lists information apars for IP-related books.

Table 64. IP Information Apars

Title	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3	TCP/IP 3.2
High Speed Access Service User's Guide (GC31-8676)	ii11629	ii11566	ii11412	ii11181		
IP API Guide (SC31-8516)	ii11635	ii11558	ii11405	ii11144		
IP CICS Sockets Guide (SC31-8518)	ii11626	ii11559	ii11406	ii11145		ii10825 ii10330
IP Configuration (SC31-8513)	ii11620	ii11555 ii11637	ii11402 ii11619	ii11159	ii10633	
IP Diagnosis (SC31-8521)	ii11628	ii11565	ii11411	ii11160 ii11414	ii10637	
IP Messages Volume 1 (SC31-8517)	ii11630	ii11562	ii11408		Messages and Codes ii10635	
IP Messages Volume 2 (SC31-8570)	ii11631	ii11563	ii11409			
IP Messages Volume 3 (SC31-8674)	ii11632	ii11564	ii11410	ii11158		
IP Migration (SC31-8512)	ii11618	ii11554	ii11401			
IP Network Print Facility (SC31-8522)	ii11627	ii11561	ii11407	ii11150		
IP Programmer's Reference (SC31-8515)	ii11634	ii11557	ii11404		ii10636	

Table 64. IP Information Apars (continued)

Title	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3	TCP/IP 3.2
IP and SNA Codes (SC31-8571)		Was created by VTAM V2R6	ii11361	ii11146		
IP User's Guide (GC31-8514)	ii11625	ii11556	ii11403	ii11143	ii10634	

Appendix J. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

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

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

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

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

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

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

Some updates of this book, such as an update between OS/390 releases, might be available only in softcopy. You can obtain softcopy from the OS/390 Online Library Collection (SK2T-6700), the OS/390 PDF Library Collection (SK2T-6718), or the OS/390 Internet Library (http://www.ibm.com/s390/os390/). To order the latest hardcopy edition that is available, you might need to order a lower suffix (dash) level.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
APPN
AS/400
AT
BookManager
BookMaster
CBPDO
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DATABASE 2
DatagLANce
DB2
DFSMS
DFSMSdfp
DFSMShsm
DFSMS/MVS
Domino
DRDA
eNetwork
Enterprise Systems Architecture/370
ESA/390
ESCON
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FAA
Micro Channel
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
MQ
Natural
NetView
Network Station
Nways
Notes
NTune
NTuneNCP
OfficeVision/MVS
OfficeVision/VM
Open Class
OpenEdition
OS/2
OS/390
Parallel Sysplex
Personal System/2
PR/SM
PROFS
PS/2
RACF
Resource Measurement Facility
RETAIN
RFM
RISC System/6000
RMF
RS/6000
S/370
S/390
SAA
SecureWay
Slate
SP
SP2
SQL/DS
System/360

FFST
FFST/2
FFST/MVS
First Failure Support Technology
GDDM
Hardware Configuration Definition
IBM
IBMLink
IMS
IMS/ESA
InfoPrint
Language Environment
LANStreamer
Library Reader
LPDA
MCS
System/370
System/390
SystemView
Tivoli
TURBOWAYS
UNIX System Services
Virtual Machine/Extended Architecture
VM/ESA
VM/XA
VSE/ESA
VTAM
WebSphere
XT
400
3090
3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

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

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/tradmarx.htm.

Bibliography

SecureWay Communications Server for OS/390 Publications

Following are descriptions of the books in the SecureWay Communications Server for OS/390 library. The books are arranged in the following categories:

- Softcopy Information
- Planning
- Resource Definition, Configuration, and Tuning
- Operation
- Customization
- Writing Application Programs
- Diagnosis
- Messages and Codes
- APPC Application Suite.
- Multiprotocol Transport Networking (MPTN) Architecture publications

The complete set of unlicensed books in this section can be ordered using a single order number, SBOF-7011.

Updates to books are available on RETAIN. See "Appendix I. Information Apars" on page 1201 for a list of the books and the INFOAPARS associated with them.

Some books are available in both hard- and soft-copy, or soft-copy only. The following abbreviations follow each order number:

HC/SC	Both hard- and soft-copy are available
SC	Only soft-copy is available

Related Publications

For information about OS/390 products, refer to *OS/390 Information Roadmap* (GC28-1727-03 [HC/SC]). The Roadmap describes what level of documents are supplied with each release of CS for OS/390, as well as describing each OS/390 publication.

Firewall

OS/390 Firewall Technologies Guide and Reference (SC24-5835-03 [HC/SC])

OSA-Express

S/390 Open Systems Adapter-Express Customer's Guide and Reference (SA22-7403-01 [HC/SC])

Softcopy Information

- *OS/390 Online Library Collection* (SK2T-6700).

  This collection contains softcopy unlicensed books for OS/390, Parallel Sysplex products, and S/390 application programs that run on OS/390. This collection is updated quarterly with any new or updated books that are available for the product libraries included in it.
- *OS/390 PDF Library Collection* (SK2T-6718).

  This collection contains the unlicensed books for OS/390 Version 2 Release 6 in Portable Document Format (PDF).
- *OS/390 Licensed Product Library* (LK2T-2499).

  This library contains unencrypted softcopy licensed books for OS/390 Version 2. If any of the books in this library are changed, it is updated quarterly. The OS/390 Licensed Product Library for Version 1 (LK2T-6702) is still available, but is no longer updated.
- *System Center Publication IBM S/390 Redbooks Collection* (SK2T-2177).

  This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

Planning

OS/390 SecureWay Communications Server: SNA Migration (SC31-8622-03 [HC/SC]). This book is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.

OS/390 SecureWay Communications Server: IP Migration (SC31-8512-03 [HC/SC]). This book is intended to help you plan for IP, whether you are migrating from a previous version or installing IP for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with IP.

Resource Definition, Configuration, and Tuning

OS/390 SecureWay Communications Server: IP Configuration (SC31-8513-03 [HC/SC]). This book is for people who want to configure, customize, administer, and maintain IP. Familiarity with MVS operating system, IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 SecureWay Communications Server: SNA Network Implementation Guide (SC31-8563-03 [HC/SC]). This book presents the major concepts involved in implementing a SNA network. Use this book in conjunction with the *OS/390 SecureWay Communications Server: SNA Resource Definition Reference*.

OS/390 SecureWay Communications Server: SNA Resource Definition Reference (SC31-8565-03 [HC/SC]). This book describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. The information includes:

- IBM-supplied default tables (logon mode and USS)
- Major node definitions
- User-defined tables and filters
- SNA start options.

Use this book in conjunction with the *OS/390 SecureWay Communications Server: SNA Network Implementation Guide*.

OS/390 eNetwork Communications Server: SNA Resource Definition Samples (SC31-8566-00 [HC/SC]). This book contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.

OS/390 eNetwork Communications Server: AnyNet SNA over TCP/IP (SC31-8578-00 [SC]). This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.

OS/390 eNetwork Communications Server: AnyNet Sockets over SNA (SC31-8577-00 [SC]). This guide provides information to help you install, configure, use, and diagnose Sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.

Operation

OS/390 SecureWay Communications Server: IP User's Guide (GC31-8514-03 [HC/SC]). This book is for people who want to use TCP/IP for data communication activities such as FTP and Telnet. Familiarity with MVS operating system and IBM Time Sharing Option (TSO) is recommended.

OS/390 SecureWay Communications Server: SNA Operation (SC31-8567-03 [HC/SC]). This book serves as a reference for programmers and operators requiring detailed information about specific operator commands.

OS/390 SecureWay Communications Server: Quick Reference (SX75-0121-03 [HC/SC]). This book contains essential information about SNA and IP operator commands.

OS/390 eNetwork Communications Server: High Speed Access Services User's Guide (GC31-8676-01 [SC]). This book is for end users and system administrators who want to use applications using a High Speed Access Services connection available in CS for OS/390.

Customization

OS/390 SecureWay Communications Server: SNA Customization (LY43-0110-01 [SC]). This book enables you to customize SNA, and includes:

- Communication network management (CNM) routing table
- Logon-interpret routine requirements
- Logon manager installation-wide exit routine for the CLU search exit
- TSO/SNA installation-wide exit routines
- SNA installation-wide exit routines

OS/390 eNetwork Communications Server: IP Network Print Facility (SC31-8522-00 [SC]). This book is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP.

Writing Application Programs

OS/390 SecureWay Communications Server: IP Application Programming Interface Guide (SC31-8516-03 [SC]). This book describes the syntax and semantics of program source code necessary to write your own application

programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this book to adapt your existing applications to communicate with each other using sockets over TCP/IP.

OS/390 SecureWay Communications Server: IP CICS Sockets Guide (SC31-8518-01 [SC]). This book is for people who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using TCP/IP for MVS.

OS/390 eNetwork Communications Server: IP IMS Sockets Guide (SC31-8519-00 [SC]). This book is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM TCP/IP for MVS.

OS/390 SecureWay Communications Server: IP Programmer's Reference (SC31-8515-03 [SC]). This book describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the MVS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 eNetwork Communications Server: SNA Programming (SC31-8573-01 [SC]). This book describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide (SC31-8581-00 [SC]). This book describes how to use the SNA LU 6.2 application programming interface for host application programs. This book applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this book.)

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Reference (SC31-8568-00 [SC]). This book provides reference material for the SNA LU 6.2 programming interface for host application programs.

OS/390 eNetwork Communications Server: CSM Guide (SC31-8575-00 [SC]). This book describes how applications use the communications storage manager.

OS/390 eNetwork Communications Server: CMIP Services and Topology Agent Guide (SC31-8576-01 [SC]). This book describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The book provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

OS/390 SecureWay Communications Server: IP Diagnosis (SC31-8521-03 [HC/SC]). This book explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.

OS/390 SecureWay Communications Server: SNA Diagnosis Volume 1: Techniques and Procedures (LY43-0079-03 [HC/SC]) and *OS/390 SecureWay Communications Server: SNA Diagnosis Volume 2: FFST Dumps and the VIT* (LY43-0080-02 [HC/SC]). These books help you identify a SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation. Volume 1 includes information about the following:

- Command syntax for running traces and collecting and analyzing dumps
- Procedures for collecting documentation (SNA, TSO)
- VIT analysis tool
- Channel programs
- Flow diagrams
- Procedures for locating buffer pools
- CPCB operation codes
- Storage and control block ID codes
- Offset names and locations for SNA buffer pools.

Volume 2 includes information about the following:

- VIT entries
- SNA internal trace
- FFST dumps and probes

OS/390 SecureWay Communications Server: Data Areas Volume 1 (LY43-0111-03 [SC]). This book

describes SNA data areas and can be used to read a SNA dump. It is intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

OS/390 SecureWay Communications Server: Data Areas Volume 2 (LY43-0112-03 [SC]). This book describes SNA data areas and can be used to read a SNA dump. It is intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and Codes

OS/390 SecureWay Communications Server: SNA Messages (SC31-8569-03 [HC/SC]). This book describes the following types of messages and other associated information:

- Messages:
  - ELM messages for logon manager
  - IKT messages for TSO/SNA
  - IST messages for SNA network operators
  - ISU messages for sockets-over-SNA
  - IVT messages for the communications storage manager
  - IUT messages
  - USS messages
- Other information that displays in SNA messages:
  - Command and RU types in SNA messages
  - Node and ID types in SNA messages
- Supplemental message-related information

OS/390 SecureWay Communications Server: IP Messages Volume 1 (EZA) (SC31-8517-03 [HC/SC]). This volume contains TCP/IP messages beginning with EZA.

OS/390 SecureWay Communications Server: IP Messages Volume 2 (EZB) (SC31-8570-03 [HC/SC]). This volume contains TCP/IP messages beginning with EZB.

OS/390 SecureWay Communications Server: IP Messages Volume 3 (EZY-EZZ-SNM) (SC31-8674-03 [HC/SC]). This volume contains TCP/IP messages beginning with EZY, EZZ, and SNM.

OS/390 SecureWay Communications Server: IP and SNA Codes (SC31-8571-03 [HC/SC]). This book describes codes and other information that display in CS for OS/390 messages.

APPC Application Suite

OS/390 eNetwork Communications Server: APPC Application Suite User's Guide (GC31-8619-00 [SC]). This book documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

OS/390 eNetwork Communications Server: APPC Application Suite Administration (SC31-8620-00 [SC]). This book contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.

OS/390 eNetwork Communications Server: APPC Application Suite Programming (SC31-8621-00 [SC]). This book provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Multiprotocol Transport Networking (MPTN) Architecture Publications

Following are selected publications for MPTN:

Networking Blueprint Executive Overview (GC31-7057)

Multiprotocol Transport Networking: Technical Overview (GC31-7073)

Multiprotocol Transport Networking: Formats (GC31-7074)

Redbooks

The following Redbooks may help you as you implement CS for OS/390.

- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 1: Configuration and Routing* (SG24–5227–01).

  This book provides examples of how to configure the base TCP/IP stack, routing daemons and the TELNET server. This book

also provides information about national language support (NLS), routing, OSPF, network interfaces, diagnosis, multicasting, OS/390 UNIX System Services and security in an OS/390 UNIX System Services environment.

- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 2: UNIX Applications* (SG24–5228–01).

  This book provides information about implementing applications that run in the OS/390 UNIX environment, such as FTP, SNMP, BIND-based name server, DHCP, and SENDMAIL. This book also provides configuration samples and describes the implementation process.

- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications* (SG24–5229–01).

  This book provides information about TCP/IP applications that run in a legacy MVS environment, including CICS/IMS Sockets, and printing (NPF, LPR, and LPD.)

- *TCP/IP in a Sysplex* (SG24–5235–01).

  The main goals of a Parallel Sysplex are high availability and high performance. This book demonstrates how these goals can be achieved in the particular environment of SecureWay Communications Server for OS/390 and its TCP/IP applications. This book describes the WLM/DNS functions, the Network Dispatcher and the Dynamic VIPA.

- *SNA and TCP/IP Integration* (SG24–5291–00).

  This book provides information about integrating current SNA network with future TCP/IP and Web-based communication requirements. This book concentrates on routing techniques.

- *SNA in a Parallel Sysplex Environment* (SG24–2113–01).

  This book provides information about implementing a VTAM-based network on a Parallel Sysplex.

- *Subarea to APPN Migration : VTAM and APPN Implementation* (SG24–4656–01).

  This book is the first of two volumes. This book provides information about the migration of a subarea network to an APPN network. Some knowledge of SNA subarea networks and familiarity with the functions, terms and data flows of APPN networks is assumed.

- *Subarea to APPN Migration : HPR and DLUR Implementation* (SG24–5204–00).

  This book is the second of two volumes. This book provides information about the coverage of a network using HPR, DLUR and APPN/HPR routers. Some knowledge of SNA subarea networks and familiarity with the functions, terms and data flows of APPN networks is assumed.

Bibliography

Index

Special Characters

/bin/ls 1153
/dev/null 1153
/dev/ptypXXXX and dev/ttypXXXX 1153
/etc/banner 1153
/etc/ftp.data 1153
/etc/gateways 1153
/etc/inetd.conf 1154
/etc/mibs.data 1154
/etc/osnmpd.data 1154
/etc/pagent.conf 1154
/etc/protocol, protocol number assignments 1193
/etc/resolv.conf 1154
/etc/resolv.conf file, configuring onslookup with 719
/etc/rsvp.conf 1154
/etc/services 1154
/etc/snmpd.conf 1154
/etc/snmptrap.dest 1154
/etc/snmpv2.conf 1154
/etc/syslog.conf 1154
.onslookuprc file, configuring onslookup with 719
'SIOCSVIPA' IOCTL 82
.TCPIP.DATA, summary of statements in 286
/usr/lib/nls/msg/C/cfmsg.cat 1154
/usr/lib/nls/msg/C/dadmsg.cat 1154
/usr/lib/nls/msg/C/dhcpsd.cat 1154
/usr/lib/nls/msg/C/ftpdmsg.cat 1157
/usr/lib/nls/msg/C/ftpdrply.cat 1157
/usr/lib/nls/msg/C/Mrm12.cat 1156
/usr/lib/nls/msg/C/msrpcgen.cat 1155
/usr/lib/nls/msg/C/msrpclib.cat 1155
/usr/lib/nls/msg/C/netmsg.cat 1157
/usr/lib/nls/msg/C/NS.cat 1156
/usr/lib/nls/msg/C/NSLK.cat 1156
/usr/lib/nls/msg/C/NSUPDATE.cat 1156
/usr/lib/nls/msg/C/omprdmsg 1157
/usr/lib/nls/msg/C/pagtsmsg.cat 1159
/usr/lib/nls/msg/C/pingmsg.cat 1157
/usr/lib/nls/msg/C/RES.cat 1156
/usr/lib/nls/msg/C/rexcmsg.cat 1157
/usr/lib/nls/msg/C/rexdmsg.cat 1158
/usr/lib/nls/msg/C/routed.cat 1158
/usr/lib/nls/msg/C/rshdmsg.cat 1158
/usr/lib/nls/msg/C/snmpclim.cat 1158
/usr/lib/nls/msg/C/snmpdmsg.cat 1158
/usr/lib/nls/msg/C/subamsg.cat 1158
/usr/lib/nls/msg/C/syslogd.cat 1155
/usr/lib/nls/msg/C/tftpdmsg.cat 1155
/usr/lib/nls/msg/C/timedmsg 1155
/usr/lib/nls/msg/C/tnmsgs.cat 1159
/usr/lib/nls/msg/C/trtemsg.cat 1159
/usr/lib/nls/msg/C/Uil12.cat 1156
/usr/lib/nls/msg/C/X11R6.cat 1157
/usr/lib/nls/msg/C/xm12.cat 1155
/usr/lib/terminfo 1159
/usr/lpp/tcpip/sample/rsvpd.conf 1160
/usr/lpp/tcpip/samples/db.data 1159
/usr/lpp/tcpip/samples/dhcpsd 1159
/usr/lpp/tcpip/samples/dnswlm.readme 1159
/usr/lpp/tcpip/samples/dpi_mvs_sample.c 1159
/usr/lpp/tcpip/samples/dpiSimpl.mi2 1159
/usr/lpp/tcpip/samples/ibm3172.mi2 1159
/usr/lpp/tcpip/samples/ibm3172.mib 1159
/usr/lpp/tcpip/samples/iwmwdnsh.h 1159
/usr/lpp/tcpip/samples/mibs.data 1159
/usr/lpp/tcpip/samples/mvstcpip.caps 1160
/usr/lpp/tcpip/samples/mvstcpip.mi2 1160
/usr/lpp/tcpip/samples/mvstcpip.mib 1160
/usr/lpp/tcpip/samples/named.boot 1160
/usr/lpp/tcpip/samples/named.lbk 1160
/usr/lpp/tcpip/samples/named.rev 1160
/usr/lpp/tcpip/samples/osnmpd.data 1160
/usr/lpp/tcpip/samples/pagent_at.conf 1160
/usr/lpp/tcpip/samples/pagent.conf 1160
/usr/lpp/tcpip/samples/pagent.ldif 1160
/usr/lpp/tcpip/samples/pagent_oc.conf 1160
/usr/lpp/tcpip/samples/protocol 1160
/usr/lpp/tcpip/samples/rfc1592b.mi2 1160
/usr/lpp/tcpip/samples/rfc1592b.mib 1160
/usr/lpp/tcpip/samples/saMIB.mi2 1160
/usr/lpp/tcpip/samples/saMIB.mib 1160
/usr/lpp/tcpip/samples/sendmail/cf 1160
/usr/lpp/tcpip/samples/sendmail/domain 1161
/usr/lpp/tcpip/samples/sendmail/feature 1161
/usr/lpp/tcpip/samples/sendmail/hack 1161
/usr/lpp/tcpip/samples/sendmail/m4 1161
/usr/lpp/tcpip/samples/sendmail/mailer 1161
/usr/lpp/tcpip/samples/sendmail/ostype 1161
/usr/lpp/tcpip/samples/sendmail/README.m4 1160
/usr/lpp/tcpip/samples/sendmail/sendmail.ps 1161
/usr/lpp/tcpip/samples/sendmail/sh 1161
/usr/lpp/tcpip/samples/sendmail/siteconfig 1161
/usr/lpp/tcpip/samples/services 1161
/usr/lpp/tcpip/samples/slapm.mi2 1160
/usr/lpp/tcpip/samples/slapm.mib 1160
/usr/lpp/tcpip/samples/slapm.txt 1160
/usr/lpp/tcpip/samples/snmpd.conf 1161
/usr/lpp/tcpip/samples/snmpv2.conf 1161
/usr/lpp/tcpip/samples/syslog.conf 1161
/usr/lpp/tcpip/samples/tcpsym 1161
/usr/lpp/tcpip/samples/wlmreg.c 1161
/usr/man/C/cat1/onetstat.1 1161
/usr/man/C/cat1/oping.1 1161
/usr/man/C/cat1/orexec.1 1161
/usr/man/C/cat1/osnmp.1 1162
/usr/man/C/cat1/otelnetd.1 1162
/usr/man/C/cat1/otracert.1 1162
/usr/man/C/cat1/uil.1 1162

Numerics

2 configuration statements'.statements, SNALINK LU
6.2 configuration
BUFFERS 351
DEST 352

2 configuration statements'.statements, SNALINK LU 6.2 configuration *(continued)*
LINK 353
TRACE 354
VTAM 355
25 configurations statements'.statements, X.25 NPSI Server configuration
ALTLINK 371
BUFFERS 373
DEST 374
FAST 375
LINK 376
OPTIONS 377
TIMERS 379
TRACE 380
VTAM 382

A

accepting the TCP/IP installation 43
access to FTP server, limiting 536
active gateway 922, 939, 1037
active route, configuring
NCPROUTE 1059
RouteD 939
ADD_NEW_KEY subcommand 884
Address Resolution Protocol (ARP) 164
alias names 640
ALTCPHOSTNAME statement 658
ALTLINK statement 371
ALTNJEDOMAIN statement (SMTP) 657
ALWAYSWTO statement 292
ANK (ADD_NEW_KEY) subcommand 884
APPL statement for SNALINK LU0 339
APPL statement for SNALINK LU6.2 348
applications, functions and protocols
Character Generator protocol 1091
Discard protocol 1091
Echo protocol 1091
File Transfer Protocol (FTP) Server 523
Kerberos 871
NCP Routing (NCPROUTE) 1035
Network Computing System (NCS) 1073
Network Database System (NDB) 1077
OROUTED Protocol 917
Portmapper 1065, 1067, 1077
Remote Execution Protocol Daemon (REXECD) 621, 625
Remote Printing 893
Remote Procedure Call (RPC)
Network Computing System (NCS) 1073
Network Data Base (NDB) 1077
Portmapper 1068
Routing Information Protocol (RIP) 193, 918, 1035
Simple Mail Transfer Protocol (SMTP) 635
Simple Network Management Protocol (SNMP) 823
SNALINK LU type 0 333
SNALINK LU type 6.2 347
X.25 NPSI Server 361
applications, search order and configuration files for the OS/390 TCP/IP 19
ARP packets 162, 163, 165
ARP table 127, 250
ARPAGE statement 120, 127
AS boundary routing, OSPF 1029
ASATRANS statement 549
ASCII-to-EBCDIC table 1135
ASSORTEDPARMS statement 128
ATCCON member of VTAMLST 40
authorize user IDs, RACF 934
automated takeover, VIPA 79
AUTOMOUNT statement
for FTP server 550
AUTOTAPEMOUNT statement 552

B

backing up an MVS host with VIPA 77
BADSPOOLFILEID statement 659
banner page 898
boot file
creating 716
sample file 797
boot file directives 792
boundary routing, OSPF 1029
bridge, token-ring 164
BSDROUTINGPARMS statement 120, 140
BUFFERS statement
SNA LU6.2 351
X.25 NPSI 373

C

CAP (CHANGE_ADMIN_PASSWORD) 884
cataloged procedures
ADM@SERV (ADM@SERV) 872
EZAFTSERV (EZAFTSRV) 524
LLBD (LLBD) 1075
MISCSERV (MISCSERV) 1093
MVSKERB (MVSKERB) 872
NDBSETUP (NDBSETUP) 1077
NRGLBD (NRGLBD) 1075
PORTC (PORTCPRC) 1082
PORTMAP (OPORTPRC) 1065, 1068
PORTS (PORTSPRC) 1081
RXSERVE (RXPROC) 621, 625
SMTP (SMTPPROC) 636
SNALINK (SNALPROC) 337
SNMPD (SNMPDPRC) 852, 853
SNMPQE (SNMPPROC) 855, 857
TCPIP (TCPIPROC) 97, 98
TCPIPL62 (LU62PROC) 348
TCPIPX25 (X25PROC) 362
CHANGE_ADMIN_PASSWORD subcommand 884
CHANGE_PASSWORD subcommand 884
channel-to-channel DEVICE and LINK statements 155
code page IBM-1047, translating to 715, 717
commands
CONVXLAT (TCP/IP) 1144
HOMETEST (TCP/IP) 41
MAKESITE (TCP/IP) 313
MODIFY (MVS)
FTP server 610

commands *(continued)*
general usage 262
NCPROUTE address space 1062
Remote Execution server 632
RouteD server 945
SNALINK LU0 342
SNALINK LU6.2 355
X.25 NPSI server 382
SMSG (TCP/IP)
general usage 284
LPD 916
SMTP 700
START (MVS) 41, 261
STOP (MVS) 261
TESTSITE (TCP/IP) 315
common configuration statements 987
communications
authentication 1173
authentication mechanism 1173
secure 1173
server authentication 1173
Component Trace 36
configuration, Dynamic VIPA 81
configuration, verifying for Dynamic VIPAs 92
configuration, VIPA takeover 81
configuration data sets
ADMADD 874
ADMGET 874
ADMMOD 874
ETCRPC 1069
HOSTS 311
KRBCONF 874
LPDDATA 897
LU62CFG 349
NPSIDATE 365
NPSIGATE 365
SAMPPROF 100, 105
SMTPCONF 648
SMTPNOTE 637
TCPDATA 287
VTAMLST
in SNALINK LU0 339
in SNALINK LU6.2 348
in X.25 NPSI 368
X25CONF 363
configuration files, how searched for 14
configuration files for the OS/390 TCP/IP applications 19
configuration files for the TCP/IP Stack 16
configuration statements, summary of TCP/IP 120
configuration statements, TCP/IP - min, max, and default parameter values 120
configuring interfaces, point-to-point 1015
Configuring Interfaces to OMPROUTE 1014
configuring OROUTED 923
configuring SNMP for OS/390 UNIX 823
configuring the FTP server 523
Configuring the OS/390 UNIX Telnet Server 387
Configuring the TFTP server 615
Configuring the TIMED daemon 617
configuring XCF 1015
connection optimization
configuring a sysplex domain
choosing sysplex name 733
configuring client applications 736
configuring for WLM registration 732, 736
configuring name servers 734
configuring WLM in goal mode 736
identifying server applications 731
updating parent name server 733
configuring a sysplex domain for
identifying name servers 733
overview 724
control characters, TCP/IP messages 56
control entries, definitions 787
conversion characters, TCP/IP messages 56
converting translation tables to binary 1144
CONVXLAT command 1136, 1144
CPW (CHANGE_PASSWORD) 884
creating a keyring file
description 1173
protecting access with SSL 1173
CS for OS/390
parts 1152
CTRACE, specifying 99
CTRACE keyword 36
customization checklist 28
customizing
general procedure 35
SMTP mail headers 638
TCP/IP messages 55

D

dadmin 1152
DATA client configuration statements, TCPIP.DATA client configuration
ALWAYSWTO 292
DATASETPREFIX 293
DOMAINORIGIN 294
HOSTNAME 295
LOADDBCSTABLES 296
MESSAGECASE 298
NSINTERADDR 299
NSPORTADDR 300
RESOLVERTIMEOUT 302
RESOLVERUDPRETRIES 303
RESOLVEVIA 301
TCPIPJOBNAME 306
TRACE RESOLVER 307
TRACE SOCKET 308
data sets 1151
data sets, overview of 14
DATASETPREFIX statement 293
DB2 connection authorization exit routine 1077
DB2 SQL
in FTP server 539
in NDB 1077
DBCS statement 660
DBCS translation tables 1138
DCB and multiple volumes 606
DDNS 738

DEBUG statement
  LPD 900
  SMTP 663
default parameter values, TCP/IP configuration statements 120
default route, configuring
  NCPROUTE 1060
  RouteD 940
default TCP/IP configuration statements 120
DELETE statement 144
demand circuit 1032
DEST statement
  FTP server 565
  SNA LU6.2 352
  X.25 NPSI server 374
DEVICE statements
  ATM 148
  channel-to-channel 155
  CLAW 151
  Ethernet link support 161
  FDDI 161, 164
  LAN Channel Station 161
  MPCIPA 169
  MPCOSA 172
  MPCPTP 175
  SNA LU Type 0 179
  SNA LU Type 6.2 182
  token ring support 161
  using the START statement 101
  virtual device 188
  X.25 NPSI Interface 185
DHCP 738
directives
  boot file 792
  resolv.conf 721
directories or files 1153
DISPLAY command
  with OMPROUTE 993
DISPLAY TCPIP,,HELP command 274
DISPLAY TCPIP,,NETSTAT command 277
DISPLAY TCPIP,,OMPROUTE command 280
DISPLAY TCPIP,,SYSPLEX command 281
DISPLAY TCPIP,,TELNET command 283
DISPLAY TCPIP command 273
domain, definition 709
domain data files
  forward
    definition 715
    sample file 794
  reverse
    definition 715
    sample file 795
Domain Name Resolution, SMTP 655
Domain Name Resolver 687
domain name system (DNS), overview 709
DOMAINORIGIN statement 294
DSN3SATH 1080
dynamic IP 737
Dynamic VIPA, configuration 81
Dynamic VIPA, resolving conflicts 86
Dynamic VIPA considerations 90
Dynamic VIPAs, configuration 91
Dynamic VIPAs, routing protocols 95
Dynamic VIPAs, verifying configuration 92
Dynamic VIPAs and UDP 90
Dynamic VIPAs within subnets 90
DYNAMICXCF 217

E

EBCDIC-to-ASCII table 1135
EBCDIC-to-ASCII translation 1133
ENDASSORTEDPARMS statement 128, 130
ENDKEEPALIVEOPTIONS statement 222
Enterprise Extender 76, 143, 175
environment, NCPROUTE 1036
ETC.SERVICES data set
  Kerberos 873
  NCPROUTE 1045
  RouteD 928
etc/services HFS file 1194
Ethernet hosts 249
Ethernet Network LCS LINK statement 162
EXIT statement 907
EXT@SRVT command, Kerberos 883
external gateway 922, 1037
external route, configuring 922
  NCPROUTE 1059
  RouteD 939
EZAVER procedure 41
EZBXFDVP utility 83

F

FAILEDJOB statement 907
FAST statement 375
fault tolerance 209
Fiber Distributed Data Interface (FDDI)
  LCS LINK statement 164
File Transfer Protocol
  anonymous logon 525
  configuration statements, PROFILE.TCPIP 523
  File Transfer Protocol (FTP) Server 523
  FTP.DATA data set 527
  limiting access to FTP server, C 536
  RACF considerations 545
  security user exits 536
  SMF configuration 533
  SMF user exit (FTPSMFEX) 535
  specifying EZAFTSRV parameters 525
  updating the FTP cataloged procedure 524
File Transfer Protocol (FTP) server, configuring 523
files 1151
filters, input/output, for RIP 920, 1039
FILTERS parameter 903
FINISHOPEN statement, SMTP 664
FTCHKCMD user exit 538
FTCHKIP user exit 537
FTCHKPWD user exit 537
FTP configuration statements
  ANONYMOUS 547
  ASATRANS 549
  AUTOMOUNT 550

FTP configuration statements *(continued)*
AUTORECALL 551
AUTOTAPEMOUNT 552
BLKSIZE 553
BUFNO 554
CCXLATE 286, 555
CHKPTINT 556
CONDDISP 557
CTRLCONN 558
DATACLASS 559
DB2 561
DB2PLAN 562
DCBDSN 563
DEST 565
DIRECTORY 566
DIRECTORYMODE 567
FILETYPE 568
INACTIVE 569
JESLRECL 570
JESPUTGETTO 571
JESRECFM 572
LRECL 573
MGMTCLASS 574
MIGRATEVOL 575
PRIMARY 576
QUOTESOVERRIDE 577
RDW 578
RECFM 579
RETPD 580
SBDATACONN 581
SECONDARY 582
SMF 583
SMFAPPE 585
SMFDEL 586
SMFEXIT 587
SMFJES 588
SMFLOGN 589
SMFREN 590
SMFRETR 591
SMFSQL 592
SMFSTOR 593
SPACETYPE 594
SPREAD 595
SQLCOL 596
STORCLASS 598
TRACE 599
TRAILINGBLANKS 600
UMASK 604
UNITNAME 605
VOLUME 606
WRAPRECORD 608
FTP server, configuring 523
FTP server files
/bin/ls 1153
FTPD 524
FTPSMFEX user exit 535
functions, summary 7

G

gateway
active 921, 922, 1037
gateway *(continued)*
external 921, 922, 1037
Interior Gateway Protocol (IGP) 918
passive 921, 1037
GATEWAY_PDS statement 1050
gateway route table name 1046
GATEWAY statement 120
NCPROUTE 1052
RouteD 929
SMTP 665
TCPIP address space 193
gateways data set
NCPROUTE 1051
RouteD 928
GET (GET_ENTRY) 885
GET_ENTRY subcommand 885

H

Hangeul DBCS translation table 1142
HFS 1151
HFS files, overview of 14
HFS files for OS/390 UNIX REXECD 621
HFS files for OS/390 UNIX RSHD 622
Hierarchical File System (HFS)
concepts 1151
high-level qualifier (HLQ) 15
hints (root server) file
definition 716
sample file 795
HLQ (high-level qualifier) 15
HOME list 207
HOME statement 207
HOMETEST 41
host name and address verification 41
HOSTNAME statement 295
HOSTS.ADDRINFO data set 316
HOSTS.LOCAL data set 310
HOSTS.SITEINFO data set 316
HYPERchannel
TRANSLATE statement 249

I

IBM 3172 Interconnect Controller 161, 162, 163
IBM 8232 LAN Channel Station 161
IBM RISC System Parallel Channel Attachment 151
ICMP 193
IDs, RACF-Authorize user 934, 956
IEFSSNxx member 100, 638
IKJTSOxx member 638
ImageServer Statement 751
in-addr.arpa domain, definition 710
INACTIVE statement 667
INCLUDE statement 211
inetd.conf, setting up 1191
initializing, NCPROUTE 1037
input/output filters, RIP 920, 1039
installing CS for OS/390 35
instances of TCPIP, considerations for multiple 57
interfaces, configuring OMPROUTE 1014
interfaces, point-to-point 1015

INTERNALCLIENTPARMS statement
with SMF 1164
InterNetwork Information Center (InterNIC) 710
IOCTL SIOCSVIPA DEFINE 257
IP addressing, virtual 71
IP forwarding 128, 214
IP routing table 193
IPCONFIG statement 120, 213
issuing multiple queries 805
ITRACE statement 220
IUCV connections
determining status 341, 359
Virtual device DEVICE and LINK 188
IUT, configuring 1015

J

Japanese DBCS translation table 1142
JES NJE node name 301

K

KADMIN command 884
KDB@DEST command, Kerberos 886
KDB@EDIT command, Kerberos 887
KDB@INIT command, Kerberos 889
KDB@UTIL command, Kerberos 890
KEEPALIVEOPTIONS statement 120, 222
Kerberos
administration 876
authentication server 876
client application 881
commands 871
configuration sample 877
configuring the server 871
data sets 873
server application 880
KSTASH command, Kerberos 891

L

LAN Channel Station DEVICE statement 161
LDAP server, service policy and 1095
limiting access to FTP server 536
LINESIZE statement 903
LINK statements
SNALINK LU6.2 353
X.25 NPSI server 376
LINK statements, TCPIP
ATM 148
channel-to-channel 155
CLAW 151
Ethernet link support 161
FDDI 161, 164
LAN Channel Station 161
MPCIPA 169
MPCOSA 172
MPCPTP 175
SNA LU Type 0 179
SNA LU Type 6.2 182
token ring support 161
LINK statements, TCPIP *(continued)*
using the START statement 101
virtual device 188
X.25 NPSI Interface 185
LIST_REQUESTS subcommand 885
LOADDBCSTABLES statement 296
LOCAL statement 903
LOCALCLASS statement, SMTP 670
LOCALDOMAIN environment variable, configuring onslookup with 719
LOCALFORMAT statement 671
LOG statement 672
LOOPBACK address 203, 207
loopback file
definition 716
sample file 797
LPD configuration statements
DEBUG 900
OBEY 902
SERVICE 903
UNIT 914
VOLUME 915
LR (LIST_REQUESTS) 885

M

MAILER statement 673
MAILFILEDSPREFIX statement, SMTP 675
MAILFILEUNIT statement, SMTP 676
MAILFILEVOLUME statement, SMTP 677
MAKESITE
batch job 314
TSO command 313
managing MVS and sysplex failures 90
maximum parameter values, TCP/IP configuration statements 120
maximum transmission unit (MTU) 140
MAXMAILBYTES statement, SMTP 678
Medium Access Control (MAC) Addresses 163, 164
MESSAGECASE statement 298
messages, TCP/IP
format 56
rules for customizing 56
messages data sets 55
metric, using to determine OSPF costs 1030
MIBDESC.DATA 857
minimum parameter values, TCP/IP configuration statements 120
MODIFY command
FTP server 610
general usage 262
NCPROUTE address space 1062
Remote Execution server 632
RouteD server 945
SNALINK LU0 342
SNALINK LU6.2 355
X.25 NPSI server 382
moving unique APF-authorized application instance (utility-activated) 89
moving unique application instance (BIND) 89
moving VIPAs manually 72
MTU 140

multiple application instance scenario, planning 84
multiple application scenario, VIPA 81
multiple copies of TCP/IP 57
multiple FTP servers 230
multiple instances of TCPIP, considerations for 57
multiple stacks
  C-INET PFS 58
  port management 58
  SMF accounting 64
  socket application programs 64
MVS Component Trace 36
MVS failure 90
MVS host
  configuring backup host with VIPA 943
  restoring primary host with VIPA 944
MVS system symbols 26
MX records 655

N

name resolution
  in a sysplex domain 725
  iterative resolution 710
name servers
  authoritative 710
  caching-only, definition 711
  configuring
    master and caching-only 714
    slaves and forwarders 714
  forwarder, definition 711
  primary, definition 711
  secondary, definition 711
  slave, definition 712
named daemon 716
  definition 717
  syntax 799
NCP host interface 1047
NCP IP router statements 1048
NCPROUTE
  building the NCPROUTE profile 1049
  catalogued procedure 1042
  configuration examples 1060
  configuring
    active route 1059
    client NCP 1045
    default route 1060
    external route 1059
    GATEWAYS data set 1051
    NCPROUTE server 1035
    passive route 1058
  filters, input/output 1039
  specifying configuration statements 1040
  updating ETC.SERVICES 1045
NCS interface
  configuration 1073
  LLBD cataloged procedure 1073
  NRGLBD cataloged procedure 1073
  specifying statements in PROFILE.TCPIP 1074
NCST (NCP Connectionless SNA Transport) 1046
NETDATA format 671
NetView 823, 858
Network DataBase System (NDB) 1077
new users, overview 7
NJE 903
  mail gateway 647
  node name 301
  with the GATEWAY statement 665
NJE statement 903
NJECLASS statement 679
NJEDOMAIN statement 680
NJEFORMAT statement 681
NOLOG statement 683
NSINTERADDR statement 299
nslookup, UNIX 803
NSPORTADDR statement 300
nsupdate 818

O

OMPROUTE, configuring 949
OMPROUTE, configuring interfaces to 1014
OMPROUTE, RACF-Authorize user IDs 956
OMPROUTE, starting 957
OMPROUTE, stopping 959
onslookup command
  command line mode
    definition 719
    options 807
    syntax 803
  interactive mode
    conditions to enter 719
    options 807
    syntax 805
  option alternatives 719
onslookup/nslookup command
  overview 719
Open Shortest Path First (OSPF) 951
operation, VIPA takeover 81
OPORTPRC 1065
OPTIONS statement 377
  NCPROUTE 1053
  OROUTED 930
orexecd 623
OROUTED 917
  configuration examples 937
  configuring the OROUTED Server 923
  filters, input/output 920
  gateways data set 928
  understanding OROUTED 917
OROUTED, configuring 923
OROUTED, starting 937
OROUTED, understanding 917
OROUTED configuration statements
  OPTIONS 930
orshd 623
OS/390 UNIX
  general considerations 35
  HFS concepts 1151
  understanding TCP/IP data set names 22
OS/390 UNIX sendmail 701
OS/390 UNIX Telnet Server, Configuring 387
osnmp command, configuring 861
OSNMPD, configuring 825
OSNMPD, starting 852

OSPF 951
areas 1023
AS boundary routing 1029
configuring 1021
configuring over ATM 1030
converting from RIP 1033
demand circuit 1032
designated router 1022
network interface parameters 1026
non-broadcast network interface parameters 1028
parameters for attached areas 1023
poll interval 1033
request hello suppression 1033
RIP comparison 1031
router IDs 1023
routing explained 1021
virtual links 1030
OSPF configuration statements 960
otelnetd 392
OUTBOUNDOPENLIMIT statement 684
overview
administration 261
customization 35
overview for new users 7

P

packet tracing 223, 343, 357, 383
PAGESIZE statement 903
parameters, FTP configuration statements
ANY, SQLCOL 596
BLOCK, SPACETYPE 594
CYLINDER, SPACETYPE 594
LABELS, SQLCOL 596
NAMES, SQLCOL 596
TRACK, SPACETYPE 594
parameters, FTP server cataloged procedure
ANONYMOUS 525
AUTOMOUNT 526
AUTORECALL 526
DATASETMODE 526
DIRECTORYMODE 526
DUMP, MODIFY 612
INACTIVE 526
JDUMP, MODIFY 612
JTRACE, MODIFY 612
NOAUTOMOUNT 526
NOAUTORECALL 526
NODUMP, MODIFY 612
NOJDUMP, MODIFY 612
NOJTRACE, MODIFY 612
NOTRACE, MODIFY 612
NOUTRACE, MODIFY 612
PORT 526
TRACE 527
TRACE, MODIFY 612
UTRACE, MODIFY 612
parameters, Kerberos
ADD_NEW_KEY, KADMIN 884
CHANGE_ADMIN_PASSWORD, KADMIN 884
CHANGE_PASSWORD, KADMIN 884
DUMP, KDB@UTIL 890
parameters, Kerberos *(continued)*
EXIT, KADMIN 885
GET_ENTRY, KADMIN 885
HELP, KADMIN 885
LIST_REQUESTS, KADMIN 885
LOAD, KDB@UTIL 890
QUIT, KADMIN 885
parameters, LPD configuration statements
CLASS 906
DEST 906
DISCARD 907
END 907
EXIT 907
FAILEDJOB 907
FILTERS 907
LINESIZE 908
LOCAL 906
MAIL 907
NJE 906
NLSTRANSLATE 909
OUTPUT 906
PAGESIZE 908
PRINTER 906
PRIORITY 906
PUNCH 906
RACF 909
REMOTE 906
SMTP 909
START 907
TRANSLATETABLE 909
XLATETABLE 909
parameters, LPD server cataloged procedure
DIAG 895
LPDDATA 895
LPDPRFX 895
TRACE 895
TYPE 895
VERSION 895
parameters, LPD SMSG interface
PRINT WORK 916
TRACE OFF 916
TRACE ON 916
parameters, Miscellaneous server
CHARGEN 1094
DEbug 1094
DISCARD 1094
ECHO 1094
TRACE 1094
parameters, NCPROUTE cataloged procedure
-dp 1044
-s 1044
-t 1044
-t-t 1044
-tq 1044
parameters, NCPROUTE gateways data set
active, gateway 939, 1052, 1053
default.router, options 1054
external, gateway 1053
host, gateway 1052
metric, gateway 1052
net, gateway 1052

parameters, NCPROUTE gateways data set *(continued)*
passive, gateway 1053
supply, options 1056
trace.level, options 1056
parameters, OROUTED cataloged procedure
-dp 935
-g 935
-q 935
-s 935
-sd 935
-sdv 935
-st 935
-sv 935
-svd 936
-t 936
-t-t 936
-t-t-t 936
-t-t-t-t 936
parameters, OROUTED gateways data set
active 929
block, options 932
external 930
forward 932
forward.cond 932
host 929
interface, options 931
interface.poll.interval, options 931
interface.scan.interval, options 931
metric, options 929
net 929
passive, options 930, 932
supply off, options 933
parameters, PORTC cataloged procedure
DB2SSID 1082
HOMEID 1082
NUMSERV 1082
TRACE 1082
USERID 1082
parameters, REXEC server
ALLCLIENTS 629
CLIENT 629
EXIT 628
LOG 629
MAXCONN 628
NOLOG 629
NOSEND 629
RACF considerations 626
RESET 629
SEND 629
TRACE 628
TSCLASS 628
TSOPROC 628
parameters, Site Table
DATACLAS, MAKESITE 313
HLQ, MAKESITE 313
MGMTCLAS, MAKESITE 313
STORCLAS, MAKESITE 313
UNIT, MAKESITE 313
VOLSER, MAKESITE 314
parameters, SMTP statements
DEBUG, SMSG 648
parameters, SMTP statements *(continued)*
EUCKANJI, DBCS 660
EXPIRE, SMSG 648
HANGEUL, DBCS 660
HELP, SMSG 648
IBMKANJI, DBCS 660
JIS78KJ, DBCS 660
JIS83KJ, DBCS 660
KSC5601, DBCS 660
LOCAL, MAILER 673
NETDATA, LOCALFORMAT 671
NETDATA, MAILER 673
NETDATA, NJEFORMAT 681
NJE, MAILER 673
NODEBUG, SMSG 648
NOLOCAL, MAILER 673
NONJE, MAILER 673
NOPRINT, REWRITE822HEADER 693
NOSOURCEROUTES, MAILER 673
NOTRACE, SMSG 648
NOUNKNOWN, MAILER 673
PRINT, REWRITE822HEADER 693
PUNCH, LOCALFORMAT 671
PUNCH, MAILER 673
PUNCH, NJEFORMAT 681
PURGE, RESTRICT 689
QUEUES, SMSG 648
RETURN, RESTRICT 689
SHUTDOWN, SMSG 648
SJISKANJI, DBCS 660
SOURCEROUTES, MAILER 673
STATS, SMSG 648
TCHINESE, DBCS 660
TRACE, SMSG 648
TRANSFERTO, RESTRICT 689
UNKNOWN, MAILER 673
parameters, SNALINK LU0
ABBREV, MODIFY 344
CLEAR, MODIFY 343
DESTPORT, MODIFY 344
FULL, MODIFY 344
HALT, MODIFY 343
IP, MODIFY 344
LINKNAME, MODIFY 343
LIST, MODIFY 343
PKTTRACE, MODIFY 343
PROT, MODIFY 344
SRCPORT, MODIFY 344
SUBNET, MODIFY 344
parameters, SNALINK LU6.2
ABBREV, MODIFY 356, 357
CANCEL, MODIFY 356
CLEAR, MODIFY 356, 357
DATA, DEST 356
DESTPORT, MODIFY 356, 357
DETAIL, TRACE 356
DROP, MODIFY 356, 357
FULL, MODIFY 356
HALT, MODIFY 356, 357
INT, DEST 356, 357
IP, MODIFY 356, 357

parameters, SNALINK LU6.2 *(continued)*
LINKNAME, MODIFY 356, 357
LIST, MODIFY 356
PKTTRACE, MODIFY 356, 357
PROT, MODIFY 356, 357
RESTART, MODIFY 356, 358
SCRPROT, MODIFY 357
SUBNET, MODIFY 356, 357
TRACE, MODIFY 356, 358
parameters, SNMP
-d, SNMP agent 858
-d, SQESERV 858
-h, SQESERV 858
-it, SQESERV 858
-s, SQESERV 858
-t, SQESERV 858
SMNPRETO, SNMPARMS 860
SNMPMMLL, SNMPARMS 860
SNMPQE, SNMPARMS 860
SNMPQERT, SNMPARMS 860
SNMPRCNT, SNMPARMS 860
SNMPRITO, SNMPARMS 860
parameters, TCPIP.DATA client statements
EUCKANJI, LOADDBCSTABLES 296
HANGEUL, LOADDBCSTABLES 296
JIS78KJ, LOADDBCSTABLES 296
JIS83KJ, LOADDBCSTABLES 296
KSC5601, LOADDBCSTABLES 297
SJISKANJI, LOADDBCSTABLES 297
TCHINESE, LOADDBCSTABLES 297
TCP, RESOLVEVIA 301
UDP, RESOLVEVIA 301
parameters, TCPIP DEVICE and LINK statements
0, LINK 153
ALLRINGSBCAST, LINK 164
CANONICAL, LINK 164
CLAW, DEVICE 151
CTC, DEVICE 155
CTC, LINK 156
FDDI, LINK 164
IBMTR, LINK 163
IP, LINK 153
LCS, DEVICE 161
LOCALBCAST, LINK 164
NONCANONICAL, LINK 164
NONE, DEVICE 151
Virtual device, DEVICE 188
Virtual device, LINK 188
parameters, TCPIP general configuration statements
ABBREV, PKTTRACE 223
CLEAR, PKTTRACE 224
DEFAULTNET, GATEWAY 194
DELAYACKS, GATEWAY 196
DELAYACKS, PORT 230
DESTPORT, PKTTRACE 224
ETHERNET, TRANSLATE 249
FALSE, BSDROUTINGPARMS 140
FDDI, TRANSLATE 249
FULL, PKTTRACE 224
HCH, TRANSLATE 249
IBMPTR, TRANSLATE 249
parameters, TCPIP general configuration statements *(continued)*
IGNOREREDIRECT, ASSORTEDPARMS 128
INTERVAL, KEEPALIVEOPTIONS 222, 247
IP, PKTTRACE 224
LINKNAME, PKTTRACE 224
MAXIMUMRETRANSMITTIME, GATEWAY 196
MINIMUMRETRANSMITTIME, GATEWAY 196
NOAUTOLOG, PORT 230
NOFWD, ASSORTEDPARMS 128
PROT, TKTTRACE 225
ROUNDTRIPGAIN, GATEWAY 196
SRCPORT, PKTTRACE 225
SUBNET, PKTTRACE 225
TRUE, BSDROUTINGPARMS 140
VARIANCEGAIN, GATEWAY 196
VARIANCEMULTIPLIER, GATEWAY 196
parameters, translation tables
CODEFILE, CONVXLAT 1144
HANGEUL, CONVXLAT 1144
KANJI, CONVXLAT 1144
TCHINESE, CONVXLAT 1144
parameters, X.25 NCP packet switching statements
ACCEPTFACILITIES, OPTIONS 377
ACCEPTREVERSE, OPTIONS 377
ASCII, TRACE 380
CALLDATA, MODIFY 377
CANCEL, MODIFY 383
CONTROL, TRACE 380
DATA, TRACE 380
DDN, LINK 376
DDN, MODIFY 371
DEBUG, MODIFY 383
EBCDIC, TRACE 380
EVENTS, MODIFY 383
FACILITIES, MODIFY 377
GATE, MODIFY 377
HALT, MODIFY 383
IA5, TRACE 380
LIST, MODIFY 383
PACKETSIZE, MODIFY 377
PKTTRACE, MODIFY 383
ABBREV 383
DESTPORT 383
FULL 383
IP 383
LINKNAME 383
PORT 383
SRCPORT 383
SUBNET 383
RESTART, MODIFY 384
REVERSE, MODIFY 377
SNAP, MODIFY 384
TRACE, MODIFY 384
TRAFFIC, MODIFY 384
WINDOWSIZE, MODIFY 377
passive gateway 921, 1037
passive route, configuring
NCPROUTE 1058
OROUTED 938
PC Network LCS LINK statement 163

performance monitoring, SLA subagent 1111
PFS 58
physical file system 58
PKTTRACE statement 120, 223
point-to-point interfaces 1015
point-to-point link, configuring (OROUTED) 940
policy agent, configuring 1095
poll interval 1033
popper 701
port assignments 1193
  /etc/services HFS file 1194
  PROFILE.TCPIP data set 1193
port ownership, specifying 714
PORT statement 120
  for SMTP 685
  TCPIP address space 229
PORTMAP address space
  configuring 1065, 1067
  starting PORTMAP 1066, 1071
  updating the PORTMAP cataloged procedure 1065, 1068
PORTRANGE statement 120
  TCPIP address space 229, 232
POSTMASTER statement, SMTP 686
PRIMARYINTERFACE statement 235
printf function 56
problem diagnosis
  checking console messages 722
  checking syslog messages 723
  using name server signals 723
  using onslookup 723
procedures, TCP/IP
  ADM@SERV (ADM@SERV) 872
  EZAFTSERV (EZAFTSRV) 524
  LLBD (LLBD) 1075
  MISCSERV (MISCSERV) 1093
  MVSKERB (MVSKERB) 872
  NDBSETUP (NDBSETUP) 1077
  NRGLBD (NRGLBD) 1075
  PORTC (PORTCPRC) 1082
  PORTMAP (OPORTPRC) 1065, 1068
  PORTS (PORTSPRC) 1081
  RXSERVE (RXPROC) 621, 625
  SMTP (SMTPPROC) 636
  SNALINK (SNALPROC) 337
  SNMPD (SNMPDPRC) 852, 853
  SNMPQE (SNMPPROC) 855, 857
  TCPIP (TCPIPROC) 97, 98
  TCPIPL62 (LU62PROC) 348
  TCPIPX25 (X25PROC) 362
PROCLIB Updates 43
PROFILE.TCPIP, specifying configuration statements
  EZAFTSRV 524
  NCPROUTE 1040
  OROUTED 925
  PORTMAP 1065, 1067
  SMTP 635
  SNALINK 334
  TCPIP 99
  X.25 NPSI 361
PROFILE.TCPIP configuration data set 119
Program Directory 35
protocol number 1193
protocol number assignments 1193
protocol suite 1183
protocols
  OSPF 1021
PUNCH format 671

Q

querying a name server 803

R

RACF
  considerations for FTP server 545
  considerations for REXEC server 626
  LPD statement 903
RACF-Authorize User IDs 934
RACF-Authorize User IDs for OMPROUTE 956
RCPT 687, 1197
RCPTRESPONSEDELAY statement, SMTP 687
registration, WLM
  overview 724
  server applications
    customized applications 736
    TCP/IP 733
    tn3270 732
remote execution server 621
Request For Comments (RFCs), obtaining 712
request hello suppression 1033
resolver configuration files 19
RESOLVERRETRYINT statement 688
resolvers, configuring host
  name server considerations 718
  onslookup considerations 722
RESOLVERTIMEOUT statement 302
RESOLVERUDPRETRIES statement 303
RESOLVEVIA statement 301
resolving conflicts, VIPA 86
resource records, definitions 787
restarting VIPADEFINE 88
restricting access to FTP server 536
RETRYAGE statement 691
RETRYINT statement 692
REWRITE822HEADER statement 693
REXECD 621, 625
REXECD, HFS files for OS/390 UNIX 621
REXECD Command 623
RIP 918, 952
  converting to OSPF 1033
  OSPF routes 1030
RIP configuration statements 976
RIP input/output filters 920, 1039
RIP_RECEIVE_CONTROL statement 1050
RIP_SUPPLY_CONTROL statement 1049
RIP2_AUTHENTICATION_KEY statement 1050
RISC/System 6000 DEVICE and LINK statements 151
Route Information Tables 1045
ROUTESA_CONFIG statement 991
routing
  OSPF 1029

Routing Information Protocol (RIP) 193, 918, 952, 1035
routing protocols, Dynamic VIPAs 95
routing table 193, 918, 1036
RPCINFO 1068
RSHD, HFS files for OS/390 UNIX 622
RSHD Command 623
RSVP agent 1121
RSVP agent, configuring 1121

S

SACONFIG statement 123, 236
SameHost, configuring 1015
sample NCP IP router statements 1048
Sample Profile Configuration Data Set (SAMPPROF) 105
SAMPPROF (Sample Profile Configuration Data Set) 105
SBCS translation tables 1133
search order for the OS/390 TCP/IP applications 19
search order for the TCP/IP stack 16
SECURE statement 694
security
  SSL 1173
sendmail, OS/390 UNIX 701
server requirements, NCPROUTE 1037
ServerType Statement 750
service level policy 1095
service policy agent, configuring 1095
SERVICE statement 903
SESSLIM parameter, VTAMLST 40
SIOCSVIPA IOCTL 255
site table 309
SLA subagent 1095
SLA subagent, configuring 1095
SLA subagent, performance monitoring 1111
SMF record layout 1163
  API calls 1165
  FTP client 1166
  FTP server 1164
  Telnet client 1167
  Telnet server 1164
SMSG interface commands
  LPD 916
  SMTP 700
SMTP configuration statements
  ALTCPHOSTNAME 658
  ALTNJEDOMAIN 657
  BADSPOOLFILEID 659
  DBCS 660
  DEBUG 663
  ENDSMSGAUTHLIST 695
  FINISHOPEN 664
  GATEWAY 665
  INACTIVE 667
  IPMAILERADDRESS 668
  LISTENONADDRESS 669
  LOCALCLASS 670
  LOCALFORMAT 671
  LOG 672
  MAILER 673
SMTP configuration statements *(continued)*
  MAILFILEDSPREFIX 675
  MAILFILEUNIT 676
  MAILFILEVOLUME 677
  MAXMAILBYTES 678
  NJECLASS 679
  NJEDOMAIN 680
  NJEFORMAT 681
  NJENODENAME 682
  NOLOG 683
  OUTBOUNDOPENLIMIT 684
  PORT 685
  PORTRANGE 232
  POSTMASTER 686
  RCPTRESPONSEDELAY 687
  RESOLVERRETRYINT 688
  RESTRICT 689
  RETRYAGE 691
  RETRYINT 692
  REWRITE822HEADER 693
  SECURE 694
  SMSGAUTHLIST 695
  SPOOLPOLLINTERVAL 696
  TEMPERRORRETRIES 697
  TIMEZONE 698
  WARNINGAGE 699
SMTP headers, customizing 638
SMTP.RULES data set 639
SMTP.SECTABLE data set 653
SNAIUCV connections
  determining status 341, 359
  Virtual device DEVICE and LINK 188
SNALINK LU6
  BUFFERS 351
  DEST 352
  LINK 353
  TRACE 354
  VTAM 355
SNALINK LU6.2
  configuration 347
  DEVICE and LINK statements 182
SNMP
  agents and subagents 825
  configuring 823
  overview 823
  query engine 857
  updating the SNMPD cataloged procedure 855
  updating the SNMPQE cataloged procedure 852
SNMP Agent (OSNMPD), configuring 825
SNMP agent (OSNMPD), starting 852
SNMP_AGENT statement 1050
SNMP_COMMUNITY statement 1050
SNMP for OS/390 UNIX, configuring 823
SNMP Subagent, configuring 866
SNMPD.CONF entries 832
SNMPIUCV module 858
SOMAXCONN statement 123, 243
special characters, definitions 791
specifying configuration statements in PROFILE.TCPIP 1040
SPOOLPOLLINTERVAL statement, SMTP 696

SQESERV module 857
SQL usage
in FTP server 539
in NDB 1077
stack affinity, specifying 714
START command 41
start procedure, updating 714
start process, configuring 717
START statement 101, 244
start the SNMP agent (OSNMPD) 852
starting
TCP/IP address space 41
TCP/IP servers 261
statements, FTP configuration
ANONYMOUS 547
ASATRANS 549
AUTOMOUNT 550
AUTORECALL 551
AUTOTAPEMOUNT 552
BLKSIZE 553
BUFNO 554
CCXLATE 286, 555
CHKPTINT 556
CONDDISP 557
CTRLCONN 558
DATACLASS 559
DB2 561
DB2PLAN 562
DCBDSN 563
DEST 565
DIRECTORY 566
DIRECTORYMODE 567
FILETYPE 568
INACTIVE 569
JESLRECL 570
JESPUTGETTO 571
JESRECFM 572
LRECL 573
MGMTCLASS 574
MIGRATEVOL 575
PRIMARY 576
QUOTESOVERRIDE 577
RDW 578
RECFM 579
RETPD 580
SBDATACONN 581
SECONDARY 582
SMF 583
SMFAPPE 585
SMFDEL 586
SMFEXIT 587
SMFJES 588
SMFLOGN 589
SMFREN 590
SMFRETR 591
SMFSQL 592
SMFSTOR 593
SPACETYPE 594
SPREAD 595
SQLCOL 596
STORCLASS 598
statements, FTP configuration *(continued)*
TRACE 599
TRAILINGBLANKS 600
UMASK 604
UNITNAME 605
VOLUME 606
WRAPRECORD 608
statements, LPD configuration
DEBUG 900
OBEY 902
SERVICE 903
UNIT 914
VOLUME 915
statements, OROUTED configuration
OPTIONS 930
statements, SMTP configuration
ALTCPHOSTNAME 658
ALTNJEDOMAIN 657
BADSPOOLFILEID 659
DBCS 660
DEBUG 663
ENDSMSGAUTHLIST 695
FINISHOPEN 664
GATEWAY 665
INACTIVE 667
IPMAILERADDRESS 668
LISTENONADDRESS 669
LOCALCLASS 670
LOCALFORMAT 671
LOG 672
MAILER 673
MAILFILEDSPREFIX 675
MAILFILEUNIT 676
MAILFILEVOLUME 677
MAXMAILBYTES 678
NJECLASS 679
NJEDOMAIN 680
NJEFORMAT 681
NJENODENAME 682
NOLOG 683
OUTBOUNDOPENLIMIT 684
PORT 685
PORTRANGE 232
POSTMASTER 686
RCPTRESPONSEDELAY 687
RESOLVERRETRYINT 688
RESTRICT 689
RETRYAGE 691
RETRYINT 692
REWRITE822HEADER 693
SECURE 694
SMSGAUTHLIST 695
SPOOLPOLLINTERVAL 696
TEMPERRORRETRIES 697
TIMEZONE 698
WARNINGAGE 699
statements, TCPIP configuration
ARPAGE 127
ASSORTEDPARMS 128
ATM DEVICE and LINK 148
BSDROUTINPARMS 140

statements, TCPIP configuration *(continued)*
channel-to-channel DEVICE and LINK 155
CLAW DEVICE and LINK 151
DELETE 144
ENDASSORTEDPARMS 128
ENDKEEPALIVEOPTIONS 222
ENDSMSGAUTHLIST 695
GATEWAY 193
HOME 207
INCLUDE 211
IPCONFIG 213
ITRACE 220
KEEPALIVEOPTIONS 222
LAN Channel DEVICE and LINK 161
MPCIPA DEVICE and LINK 169
MPCOSA DEVICE and LINK 172
MPCPTP DEVICE and LINK 175
PKTTRACE 223
PORT 229
PORTRANGE 232
PRIMARYINTERFACE 235
SACONFIG 236
SNA LU Type 6.2 DEVICE and LINK 182
SOMAXCONN 243
START 244
STOP 246
TCPCONFIG 247
TRANSLATE 249
UDPCONFIG 252
Virtual Device DEVICE and LINK 188
X.25 NPSI DEVICE and LINK 185
statements, Telnet server configuration
ENDVTAM 355
statements in TCPIP.DATA, summary 286
STOP command 261
STOP statement 246
stopping TCP/IP 261
subagent, SLA performance monitoring 1111
subnet masks 194
subnets, Dynamic VIPAs within 90
summary of functions 7
summary of statements in TCPIP.DATA 286
symbols, MVS system 26
syslog file, creating 718
syslogd
command format 1184
exit values 1185
files used by 1184, 1185
overview 1183
starting 1184
stopping 1185
sysplex, definition 724
sysplex failure 90
system parameters for clients 285
system symbols, MVS 26

T

task of service 1095
TCP/IP
resolvers 19
TCP/IP *(continued)*
understanding data set names with OS/390 UNIX services 22
TCP/IP, stopping 261
TCP/IP Cataloged Procedure (TCPIPROC) 97
TCP/IP Cataloged Procedure, Updating the 97
TCP/IP configuration statements - min, max, and default parameter values 120
TCP/IP for MVS
configuring 97
starting a TCPIP server 36
TCP/IP stack, search order and configuration files for the 16
TCPCONFIG statement 123, 247
TCPDATA 287
TCPIP
ALWAYSWTO 292
DATASETPREFIX 293
DOMAINORIGIN 294
HOSTNAME 295
LOADDBCSTABLES 296
MESSAGECASE 298
NSINTERADDR 299
NSPORTADDR 300
RESOLVERTIMEOUT 302
RESOLVERUDPRETRIES 303
RESOLVEVIA 301
TCPIPJOBNAME 306
TRACE RESOLVER 307
TRACE SOCKET 308
TCPIP configuration statements, summary of 120
TCPIP configurations statements
ARPAGE 127
ASSORTEDPARMS 128
ATM DEVICE and LINK 148
BSDROUTINPARMS 140
channel-to-channel DEVICE and LINK 155
CLAW DEVICE and LINK 151
DELETE 144
ENDASSORTEDPARMS 128
ENDKEEPALIVEOPTIONS 222
ENDSMSGAUTHLIST 695
GATEWAY 193
HOME 207
INCLUDE 211
IPCONFIG 213
ITRACE 220
KEEPALIVEOPTIONS 222
LAN Channel DEVICE and LINK 161
MPCIPA DEVICE and LINK 169
MPCOSA DEVICE and LINK 172
MPCPTP DEVICE and LINK 175
PKTTRACE 223
PORT 229
PORTRANGE 232
PRIMARYINTERFACE 235
SACONFIG 236
SNA LU Type 6.2 DEVICE and LINK 182
SOMAXCONN 243
START 244
STOP 246

TCPIP configurations statements *(continued)*
TCPCONFIG 247
TRANSLATE 249
UDPCONFIG 252
Virtual Device DEVICE and LINK 188
X.25 NPSI DEVICE and LINK 185
TCPIP.DATA data set, configuring onslookup with 719
TCPIPJOBNAME statement 306
TCPIPROC (TCP/IP Cataloged Procedure) 97
Telnet 3270 DBCS transform mode 1140
Telnet configuration statements
ENDVTAM 355
Telnet Server, Configuring 387
TEMPERRORETRIES statement 697
test configuration port assignments 1193
TESTSITE 315
TFTP server
configuring 615
TIMED daemon
configuring 617
TIMERS statement 379
TIMEZONE statement, SMTP 698
TNF 37
token-ring
bridge 164
hosts 249
LCS LINK statement 163
TRACE RESOLVER command 307
TRACE SOCKET command 308
TRACE Statement
FTP server 599, 600
SNA LU6.2 354
X.25 NPSI 380
Traditional Chinese DBCS translation table 1143
Traffic, splitting with VIPA 941
TRAILINGBLANKS statement 600
TRANSLATE statement 249
TRANSLATETABLE statement 903
translating boot files 717
translating data files 715
translating TCP/IP messages 55
translation tables
converting to binary 1144
CONVXLAT command 1136, 1144
DBCS 1138
loading 296
SBCS 1133
using 1133

U

UDP 128, 232
UDPCONFIG statement 123, 252
understanding TCP/IP data set names with OS/390 UNIX services 22
understanding virtual IP addressing 71
unique application instance scenario, planning 85
unique application scenario, VIPA 82
UNIT statement 914
UNIX nslookup 803
user IDs, RACF-Authorize 934, 956
users, overview for new 7

V

VARY TCPIP,,DROP command 268
VARY TCPIP,,OBEYFILE command 267
VARY TCPIP,,PKTTRACE command 271
VARY TCPIP,,START command 269
VARY TCPIP,,STOP command 269
VARY TCPIP,,TELNET command 270
VARY TCPIP command 264
verification
host name and address 41
system configuration 40
X Window System 42
VIPA, introduction 71
VIPA, manual movement 72
VIPA, overview 71
VIPA takeover, configuration and operation 81
VIPA takeover, planning 79
VIPADEFINE, restart 88
virtual device
example of BSDROUTINGPARMS definitions 142
virtual IP address support (VIPA)
backing up MVS host 77
configuration example 209
configuring
backup MVS host 943
on HOME statement 208
primary MVS host 944
splitting traffic with 941
virtual IP addressing, overview 71
virtual machine communication facility, see also VMCF 37
VMCF (virtual machine communication facility)
commands 38
configuring as non-restartable system 38
configuring as restartable system 37
VTAM APPL definition
for SNALINK LU0 339
for SNALINK LU6.2 348
for X.25 NPSI 368
VTAM configuration statements
SNA LU6.2 355
X.25 NPSI 382
VTAM parameters, general update 40
VTAM statement 915
SNALINK LU 6.2 355
X.25 382
VTAMLST member 40

W

WARNINGAGE statement 699
well-known procedure names, defining 1068

X

X
ALTLINK 371
BUFFERS 373

X *(continued)*
DEST 374
FAST 375
LINK 376
OPTIONS 377
TIMERS 379
TRACE 380
VTAM 382
X Window System verification 42
XCF, configuring 1015
XLATETABLE statement 903

Z

zone transfers 711

Readers' Comments — We'd Like to Hear from You

OS/390 SecureWay Communications Server
IP Configuration
Version 2 Release 8

Publication No. SC31-8513-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-8513-03

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC
27709-9990

Fold and Tape **Please do not staple** Fold and Tape

SC31-8513-03

Cut or Fold
Along Line

File Number: S390-50
Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8513-03